НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.91 : 378.14

«До захисту допущено»
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
“___”_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 126 «Інформаційні системи та технології»

на тему: Гнучкий роботизований сервіс розрахунку академічної заборгованості з надання освітніх послуг

Виконав: студент другого курсу, групи ІК-82мп
(шифр групи)

Коломоєць Сергій Олексійович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н., доцент, Остапченко К.Б. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант НК к.т.н., доцент, Пасько В.П. _____
(назва розділу) (науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент каф. АСОІУ, к.т.н., доцент, Муха І.П. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ І.Р. Пархомей
(підпис)
«___»_____2019 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Коломойцю Сергію Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Гнучкий роботизований сервіс розрахунку академічної заборгованості з надання освітніх послуг

науковий керівник дисертації Остапченко Костянтин Борисович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «28» 10 2019 р. № 3770-с

2. Термін подання студентом дисертації 18.11.2019

3. Об'єкт дослідження розрахунок академічної заборгованості з надання освітніх послуг в вищих навчальних закладах

4. Предмет дослідження методи автоматизації розрахунку академічної заборгованості в вищих навчальних закладах

5. Перелік завдань, які потрібно розробити 1. Аналіз існуючих рішень
2. Створення алгоритму автоматизації
3. Створення додатку розрахунку академічної заборгованості з надання освітніх послуг у вищих навчальних закладах

6. Орієнтовний перелік ілюстративного матеріалу _____

_____

_____

_____

_____

7. Орієнтовний перелік публікацій — 2 публікації.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
НК	Пасько В.П., доцент		
Перевірка на співпадіння	Лісовиченко О.І., доцент		

9. Дата видачі завдання 01.09.2018 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Формування проблематики	02.09.2019 – 08.09.2019	
2	Аналіз проблематики	09.09.2019 – 15.09.2019	
3	Постановка задачі	16.09.2019 – 22.09.2019	
4	Аналітичний підхід до вирішення задач	23.09.2019 – 29.09.2019	
5	Вибір технології розробки ПЗ	30.09.2019 – 06.10.2019	
6	Розробка ПЗ	07.10.2019 – 13.10.2019	
7	Тестування та покращення ПЗ	14.10.2019 – 20.10.2019	
8	Практичне застосування ПЗ	21.10.2019 – 27.10.2019	

Студент _____ Коломоєць С.О.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Остапченко К.Б.
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

У магістерській дисертації розглянуто проблему розрахунку академічної заборгованості з надання освітніх послуг у вищих навчальних закладах.

У розділі проблематики було розглянуто академічну заборгованість з надання освітніх послуг у вищих навчальних закладах, електронний документообіг, переваги та недоліки існуючих аналогів та сформульовано необхідність створення нової системи.

У розділі вибору технологій розробки розглянуто сучасні операційні системи, мови програмування та програмні рішення для розробки власного програмного забезпечення. Визначено вимоги до системи.

У розділі практичного застосування наведено приклади роботи сервісу та описується процес взаємодії користувача з додатком.

У розділі маркетингового аналізу стартап-проекту проаналізовано поточну ситуацію на ринку, розроблено стратегії та маркетинговий плани для впровадження даного рішення.

Ключові слова: академічна заборгованість, надання освітніх послуг, електронний документообіг, прикладний програмний інтерфейс.

Розмір пояснювальної записки — 70 сторінок містять 46 ілюстрацій, 24 таблиць.

ABSTRACT

The master`s thesis deals with the problem of calculating academic debt for the provision of educational services in higher education institutions.

The section dealt with the academic debt on the provision of educational services in higher education institutions, electronic workflows, advantages and disadvantages of existing analogues, and formulated the need for a new system.

The section on the choice of development technologies examines modern operating systems, programming languages and software solutions for the development of their own software. System requirements were defined.

The practical application section provides examples of how the service works and describes how the user interacts with the application.

The Marketing Analysis section of the start-up project analyses the current market situation, develops strategies and marketing plans to implement this solution.

Key words: academic debt, provision of educational services, electronic document flow, application programming interface.

Explanatory note size — 70 pages contain 46 illustrations, 24 tables.

Пояснювальна записка
до магістерської дисертації

на тему: Гнучкий роботизований сервіс розрахунку академічної заборгованості з надання освітніх послуг

Київ – 2019 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

АЗ — академічна заборгованість

БД — база даних

ВНЗ — вищий навчальний заклад

ГРС — гнучкий роботизований сервіс

ЕК — екзаменаційна комісія

НОП — надання освітніх послуг

ОС — операційна система

ПЗ — програмне забезпечення

РС — роботизований сервіс

СЕД — система електронного документообігу

API (application programming interface) — прикладний програмний інтерфейс

ВСТУП

Актуальність даної дисертації полягає у автоматизації розрахунку академічної заборгованості, що дозволить зменшити використання людських ресурсів до мінімуму, за допомогою кросплатформленого додатку та дозволить зменшити навантаження співробітників ВНЗ для роботи, яка не може бути автоматизованою. На сьогоднішній день згідно відкритих джерел інформації не існує програмних рішень проблеми з автоматизації розрахунку академічної заборгованості, що підвищує її значимість.

Об’єктом у магістерській дисертації виступає розрахунок академічної заборгованості з надання освітніх послуг, предметом — методи проектування академічної заборгованості.

Наукова новизна даної магістерської дисертації полягає у застосуванні удосконалених методів, методологій та методик для вирішення проблеми розрахунку академічної заборгованості. Практична новизна полягає у врахуванні всіх критеріїв для повноцінної роботи з академічною заборгованістю.

Для даної магістерської дисертації було поставлено наступні задачі:

1. провести аналіз існуючих систем та методів роботизованих систем для ВНЗ;
2. сформулювати вимоги до системи управління;
3. адаптувати систему управління для роботи у структурах з надання освітніх послуг.

Практична цінність магістерської дисертації полягає в автоматизації процесу керування академічною заборгованістю.

У рамках магістерської дисертації було написано 2 статті. Перша стаття була опублікована у науковому журналі International Electronic Scientific and Practical Journal «WayScience» №2 (4) 2019 та пройшла апробацію на IX Міжнародній науково-практичній інтернет-конференції «Сучасний рух науки» м.Дніпро (Україна). Друга стаття була опублікована у збірнику XV

Міжнародна науково-практична конференція «Перспективні питання світової науки — 2019» та пройшла апробацію на науково-практичній конференції у м.Софія (Болгарія).

РОЗДІЛ 1. АНАЛІЗ РІВНЯ АВТОМАТИЗАЦІЇ ОБЛІКУ АКАДЕМІЧНОЇ ЗАБОРГОВАНОСТІ З НАДАННЯ ОСВІТНІХ ПОСЛУГ

1.1 Електронний документообіг

Автоматизація документів — це проектування систем та робочих процесів, що сприяють створенню електронних документів [1]. Сюди входять системи, засновані на логіці, які використовують сегменти вже існуючого тексту та/або даних для збирання нового документа. Цей процес все частіше використовується в певних галузях промисловості для збирання юридичних документів, контрактів та листів. Системи автоматизації документів можуть також використовуватися для автоматизації всього умовного тексту, змінного тексту та даних, що містяться в наборі документів [2].

Системи автоматизації дозволяють компаніям мінімізувати введення даних, скоротити витрачений час на перевірку читання та зменшити ризики, пов’язані з помилка-ми людини [3]. Додаткові переваги включають: економію часу та фінансових ресурсів через зменшення роботи з папером, завантаження документів, зберігання, розповсюдження, поштове відправлення або достав-ку, факси, телефон, робочу силу та відходи.

Існують чітко визначені компоненти автоматизації документообігу.

1. Метадані. Метадані зазвичай зберігаються для кожного документа. Метадані можуть, наприклад, включати дату збереження документа та особу користувача, який його зберігає. DMS також може автоматично витягувати метадані з документа або спонукати користувача додавати метадані. Деякі системи також використовують оптичне розпізнавання символів на відсканованих зображеннях або виконують вилучення тексту на електронних документах. Отриманий витягнутий текст може бути використаний для надання допомоги користувачам у пошуку документів шляхом виявлення ймовірних ключових слів або надання можливості повнотекстового пошуку тексту, або може використовуватися

самостійно. Витягнутий текст також може зберігатися як складова метаданих, зберігатися разом із документом або окремо від документа як джерело для пошуку колекцій документів.

2. Інтеграція. Багато систем управління документами намагаються надати функціонал управління документами безпосередньо іншим програмам, так що користувачі можуть отримувати наявні документи безпосередньо з сховища системи управління документами, вносити зміни та зберігати змінений документ назад у сховище як нову версію, все без виходу додаток. Така інтеграція зазвичай доступна для різних програмних засобів, таким як управління робочим процесом та системами управління контентом, як правило, через інтерфейс програмування прикладних програм (API).
3. Захоплення. Захоплення в першу чергу передбачає прийняття та обробку зображень паперових документів зі сканерів або багатофункціональних принтерів. Програмне забезпечення для оптичного розпізнавання символів (OCR) часто використовується, інтегроване в апаратне забезпечення або як автономне програмне забезпечення, щоб перетворити цифрові зображення в машиночитаний текст. Програмне забезпечення для оптичного розпізнавання знаків (OMR) іноді використовується для отримання значень прапорців або бульбашок. Захоплення може також включати прийняття електронних документів та інших файлів на комп’ютері.
4. Правила перевірки даних. Правила перевірки даних можуть перевірити наявність несправностей документа, відсутні підписи, неправильно написані імена та інші проблеми, рекомендуючи параметри виправлення в реальному часі перед імпортом даних у DMS. Додаткова обробка у вигляді гармонізації та зміни формату даних також може застосовуватися як частина перевірки даних.

5. Індексація. Індексація відстежує електронні документи. Індексація може бути такою ж простою, як і відстеження унікальних ідентифікаторів документів; але часто він має більш складну форму, забезпечуючи класифікацію за допомогою метаданих документів або навіть через індекси слів, витягнуті зі змісту документів. Індексація існує в основному для підтримки інформаційного запиту та пошуку. Однією з важливих для швидкого пошуку напрямків є створення індексної топології чи схеми.
6. Зберігання електронних документів. Зберігання документів часто включає управління тими ж документами; де вони зберігаються, як довго триває переміщення документів з одного носія інформації на інший (ієрархічне управління зберіганням) та можливе знищення документів.
7. Пошук. Поняття отримання певного документа є простим, пошук в електронному контексті може бути досить складним і потужним. Просте отримання окремих документів може бути підтримане, дозволяючи користувачеві вказати унікальний ідентифікатор документа та змусити систему використовувати базовий індекс (або неіндексований запит у своєму сховищі даних) для отримання документа. Більш гнучкий пошук дозволяє користувачеві вказати часткові пошукові терміни, що включають ідентифікатор документа та/або частини очікуваних метаданих. Зазвичай це повертає список документів, які відповідають пошуковим умовам користувача. Деякі системи надають можливість задати булевий вираз, що містить кілька ключових слів або прикладів фраз, які, як очікується, будуть міститись у вмісті документів. Визначення цього запиту може підтримуватися попередньо вбудованими індексами або може здійснювати більш трудомісткі пошуки через вміст документів, щоб повернути список потенційно відповідних документів.

8. Поширення. Документ, готовий до розповсюдження, повинен бути у форматі, який неможливо легко змінити. Оригінальна головна копія документа зазвичай ніколи не використовується для розповсюдження; швидше, електронне посилання на сам документ є більш поширеним. Якщо документ повинен бути розповсюджений в електронному вигляді в регуляторному середовищі, необхідно дотримуватися додаткових критеріїв, включаючи гарантії простежуваності та версій, навіть у інших системах. Цей підхід застосовується до обох систем, за допомогою яких документ повинен бути обмінений , якщо цілісність документа є обов'язковою.
9. Безпека. Безпека документів є життєво важливою для багатьох програм управління документами. Вимоги дотримання певних документів можуть бути досить складними залежно від типу документів. Наприклад, у Сполучених Штатах такі стандарти, як ISO 9001 та ISO 13485, а також американські постанови щодо управління харчовими продуктами та лікарськими препаратами диктують, як слід вирішувати процес контролю над документами. Системи управління документами можуть мати модуль управління правами, який дозволяє адміністратору надавати доступ до документів на основі типу лише певним людям або групам людей. Маркування документів під час друку чи створення PDF - це важливий елемент для запобігання внесенню змін або випадкового використання.
10. Робочий процес. Робочий процес є складним процесом, і деякі системи управління документами мають або вбудований модуль робочого процесу, або можуть інтегруватися з інструментами управління робочим потоком. Існують різні типи робочого процесу. Використання залежить від середовища, до якого застосовується електронна система управління документами (СУД). Ручний робочий процес вимагає від користувача перегляду документа та вирішення,

до кого його надсилати. Робочий процес на основі правил дозволяє адміністратору створити правило, яке диктує потік документа через організацію: наприклад, рахунок-фактура проходить через процес затвердження, а потім направляється в дебіторську заборгованість відділу. Динамічні правила дозволяють створювати гілки в процесі робочого процесу. Простим прикладом може бути введення суми рахунка-фактури, і якщо сума нижча за певну встановлену суму, вона слід за різними маршрутами через організацію. Розширені механізми робочого процесу можуть керувати вмістом або сигналом зовнішніх процесів, поки ці правила діють.

11. Співпраця. Співпраця має бути притаманною СЕД. У своїй базовій формі спільна СЕД повинна дозволяти відбирати документи та працювати з ними авторизованим користувачем. Доступ повинен бути заблокований іншим користувачам під час роботи над документом. Інші вдосконалені форми співпраці діють у режимі реального часу, дозволяючи багатьом користувачам одночасно переглядати та змінювати (або розмічувати) документи. Отриманий документ є вичерпним, включаючи всі доповнення користувачів. Співпраця в системах управління документами означає, що різні розмітки кожного окремого користувача під час сеансу співпраці записуються, що дозволяє стежити за історією документів.
12. Версіювання. Версіювання — це процес, за допомогою якого документи перевіряються системою управління документами, що дозволяє користувачам отримувати попередні версії та продовжувати роботу з обраної точки. Версія корисна для документів, які змінюються в часі і потребують оновлення, але, можливо, доведеться повернутися до попередньої копії або посилатися на неї.

13. Одночасний пошук. Це стосується можливості розширення можливостей пошуку для отримання результатів з декількох джерел або з декількох DMS в межах підприємства.
14. Публікація. Публікація документа включає процедури коректури, експертного чи громадського огляду, дозволу, друку та затвердження тощо. Ці кроки забезпечують розсудливість та логічне мислення. Будь-яке необережне поводження може призвести до неточності документа і, таким чином, ввести в оману або засмутити його користувачів та читачів. У законодавчих галузях, що регулюються законодавством, деякі процедури повинні бути завершені, як це засвідчено відповідними підписами та датою підписання документа. Опублікований документ повинен бути у форматі, який не легко змінити без конкретних знань чи інструментів, і все ж він є лише для читання або для перенесення.
15. Відтворення на паперовому носії. Відтворення документа або зображення часто необхідне в системі управління документами, і слід підтримувати його підтримувані пристрої виводу даних та можливості відтворення.

1.2 Особливості процесу надання освітніх послуг

Згідно чинного закону України № 2145-VIII Про освіту від 09.08.2019 року, освітні послуги — це комплекс визначених законодавством, освітньою програмою та/або договором дій суб’єкта освітньої діяльності, що мають визначену вартість та спрямовані на досягнення здобувачем освіти очікуваних результатів навчання [4].

Надання освітніх послуг — діяльність підприємств, установ, організацій, громадян (суб’єктів господарювання) та окремих осіб (викладачів, майстрів, тренерів, вихователів, репетиторів), виконувана для задоволення потреб людей (учнів, вихованців, студентів, аспірантів, докторантів), роботодавців та держави, інших громадян та суб’єктів

господарювання з передачі протягом певного часу або постійно сукупності знань, умінь та навичок, що визначають певний їх рівень або ступінь, інших прав, належних НЗ, не має матеріальної форми, не залежить від характеру результату, на платній чи безоплатній основі, що має цінову визначеність.

Академічна заборгованість — заборгованість, яка виникає якщо здобувач вищої освіти протягом семестру не виконав умов визначених навчальним планом з будь-якої навчальної дисципліни через низький рівень знань, недбале ставлення до навчання або пропуски занять. У разі, коли у визначений строк академічну заборгованість не ліквідовано, здобувач вищої освіти підлягає відрахуванню.

Для поновлення після відрахування студент повинен пройти ту саму програму за семестр або рік, в залежності від моменту виключення, склавши залік або екзамен з дисциплін з яких він має академічну заборгованість. Розрахунок академічної заборгованості з надання освітніх послуг повинен надаватися деканатом факультету, де навчався та збирається продовжувати навчання студент, у вигляді кількох документів: договору про надання освітніх послуг, акту виконаних робіт, розрахунку освітніх послуг.

На сьогоднішній день вся робота з розрахунку академічної заборгованості з надання освітніх послуг виконується працівниками деканатів ВНЗ, що призводить до додаткового навантаження співробітників ВНЗ та неефективного розподілу ресурсів.

На сьогоднішній день відбувається перехід від звичної форми праці до її автоматизації, що в свою чергу економить такі ресурси, як час, кошти та робить оптимізацію використання людських ресурсів. Але через те, що галузь освіти направлена на НОП, бюрократичний апарат не має достатнього державного фінансування. Звичайно що технічні ВНЗ мають доступ до безкоштовних підписок на всі продукти для студентів на момент їх навчання від компаній як Microsoft, або суттєві знижки від компаній з перевірки робіт на плагіат як Unicheck, а також спеціальні ціні на сервіси за підпискою від

компаній як Apple, але коли справа стосується бюрократичної частини, то вся робота, зазвичай, виконується працівниками ВНЗ. У сьогоднішніх реаліях новий уряд оголосив курс на автоматизацію управління, спрощення бюрократії та знайдення нових шляхів в управлінні різноманітними структурами, в якій сфера освіти потребує не меншої уваги ніж усі інші [5].

Проаналізувавши діяльність Міністерства освіти і науки України, а також найбільших ВНЗ було зроблено висновок, що станом на сьогодні питання автоматизації управління керуючими апаратами відповідних структур, якщо і вирішується, то лише на «локальному» рівні, що робить досвід окремих структур не доступними для інших і неможливо певно стверджувати, що взагалі існують програмні рішення для розв'язання задачі автоматизації. Не існує єдиного стандартизованого рішення для питання розрахунку академічної заборгованості.

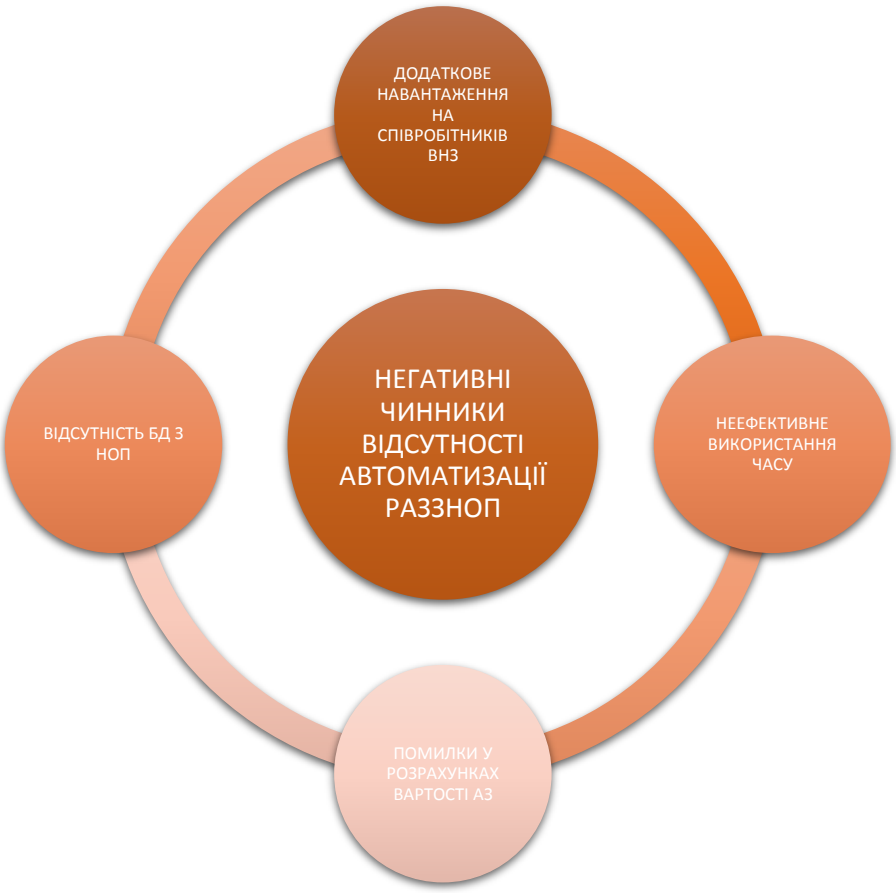


Рисунок 1.1. Негативні чинники відсутності автоматизації академічної заборгованості

Для «повноцінного» електронного урядування у сфері НОП потрібно вирішити наступні задачі:

- розрахунок стипендій та заробітних плат співробітників ВНЗ;
- розрахунок академічної заборгованості;
- формування наказів, дипломів;
- використання електронних довірчих послуг;

1.3 Аналіз останніх досліджень і публікацій

На сьогоднішній день існує велика кількість сервісів для ведення електронного документообігу [6]. В Україні є велика кількість пропозицій на цьому ринку, тому існують кілька десятків компаній, які пропонують свої послуги. Побажання користувача будуть виконані у повній мірі, тому що через велику конкурентність існує різноманіття. Вибирати можна від функціоналу, інтерфейсу до цін, а завдяки реформуванні законодавства та намірам нового уряду майже повністю перейти на електронний документообіг, ринок буде тільки збільшуватись.

На сьогоднішній день існує багато сервісів електронного документообігу в Україні:

1. Microsoft SharePoint
2. FossDoc
3. XPages Dynamic
4. DOCS.UA
5. Megapolis
6. Directum
7. elDoc
8. АСКОД
9. Alfresco
10. DocsVision
11. ТЕЗИС
12. Star.Docs (Київстар)

13.Mobile ID (lifecell)

14.Paperless (Приват Банк)

Але у галузі надання освітніх послуг питання електронного документообігу зовсім не розкрите. Співробітникам деканатів приходиться виконувати усю рутинну роботу вручну. Цей факт сповільнює функціонування апарату ВНЗ та коефіцієнт корисної дії тримається на низькому рівні.

Згідно відкритої інформації, на сьогоднішній день в Україні не існує роботизованих сервісів, які б дозволили автоматизувати управління ВНЗ. Якщо подібні рішення і існують, то лише у межах певного ВНЗ, що робить їх недоступними для загального аналізу та використання. Зазвичай подібними продуктами не прийнято ділитися, тому що вони відповідають вимогам лише окремо взятого навчального закладу та зроблені на основі знань студентів що там навчаються і не мають комерційної складової та не пристосовані для загального використання, або навчальний заклад зовсім не має засобів для автоматизації документообігу [7].

1.4 Постановка задачі автоматизації процесів розрахунку академічної заборгованості з надання освітніх послуг

Існує чітко визначений процес за яким відбувається ліквідація академічної заборгованості (рис.1): виникнення заборгованості, виключення, формування документів на поновлення, ліквідація академічної заборгованості, продовження/закінчення студентом навчального процесу. Виникнення заборгованості може відбуватися у разі отримання оцінки «незадовільно», не допуску до підсумкового контролю за результатами поточного оцінювання, студент не з’явився на екзамен без надання виправдовуючого документа. Формування документів на поновлення здійснюється деканатом (інколи кафедрою) факультету на якому навчався студент, саме цей процес розглядається у магістерській дисертації. Ліквідація АЗ відбувається шляхом повторного вивчення дисципліни та перезаліку у вигляді заліку або екзамену

(або якщо студент навчається на останньому семестрі магістратури, то перескладання буде лише захисту магістерської дисертації перед екзаменаційною комісією). Продовження/закінчення студентом навчального процесу відбувається у разі складанням усіх норм навчального плану.

Рисунок 1.2. Порядок ліквідації академічної заборгованості

Висновки до розділу

Відсутність розрахунку академічної заборгованості — одна з основних проблем у навчальному процесі.

Не існує готових та стандартизованих програмних рішень для вирішення питання розрахунку академічної заборгованості згідно відкритих джерел інформації.

Автоматизація розрахунку академічної заборгованості дозволить вирішити ряд питань: додаткового навантаження на співробітників ВНЗ, неефективного використання часу, надійного зберігання БД студентів з академічною заборгованістю яким були надані освітні послуги, помилок у розрахунку вартості академічної заборгованості.

РОЗДІЛ 2. ОБГРУНТУВАННЯ ОБРАНИХ ТЕХНОЛОГІЙ

2.1 Конфігурація технічного та програмного забезпечення процесів надання освітніх послуг ВНЗ України

Практичним способом досліджено технічне та програмне забезпечення «КПІ ім. Ігоря Сікорського» на прикладі Факультету інформатики та обчислювальної техніки (кафедра технічної кібернетики) та зроблено експертний висновок про стан техніко-програмної бази ВНЗ. Спеціалізовані аудиторії для комп'ютерних практикумів обладнано ПК різних брендів, на всіх пристроях установлена ОС Windows, найнижча версія операційної системи — Windows 7. Інші популярні ОС (наприклад Linux або Macintosh) не використовувались.

Проаналізувавши технічне та програмне забезпечення ВНЗ України і сконцентрувавши увагу саме на «КПІ ім. Ігоря Сікорського» було зроблено наступні висновки:

- усіма ВНЗ використовується операційна система Windows;
- операційні системи Linux та Macintosh не використовуються на рівні ВНЗ взагалі (винятками можуть бути лише персональні комп'ютери викладачів, що ніяк не впливає на освітній процес, який зазвичай виконується за допомогою робочого обладнання);
- зазвичай використовується версія Windows 7 (у деяких випадках, якщо ВНЗ не має змоги перейти на сьому версію використовують Vista або XP, але це скоріше виняток ніж закономірність);
- потрібно забезпечити сумісність додатку з новішими версіями операційної системи як Windows 8/8,1/10;
- для роботи з документообігом використовують пакет Microsoft Office (визначення конкретної версії якою користуються ВНЗ є неможливим, тому потрібно зробити так щоб підтримка була оптимальною для якомога більшої кількості версій MS Office);

- робота з розробленим додатком має бути без додаткового завантаження стороннього програмного забезпечення;

Було сформульовано критерії, за якими повинен бути розроблений додаток:

- користувач (співробітник деканату) повинен мати змогу власноруч змінювати параметр у налаштуваннях вартості надання послуги академічної заборгованості за науковим ступенем викладача (вартість може змінюватись з кожним роком, але це не повинно робити ПЗ «життєпридатним» лише для умов, які існують на сьогоднішній день);
- користувач (співробітник деканату) повинен мати змогу власноруч змінювати параметр у налаштуваннях норми часу для студентів 1-2 курсів та 3-6 курсів (вартість може змінюватись з кожним роком, але це не повинно робити ПЗ «життєпридатним» лише для умов, які існують на сьогоднішній день);
- у налаштуваннях повинен бути пункт шляху збереження файлу, який користувач може змінювати;
- збереження параметрів в налаштуваннях повинно ставати за замовченням щоб користувач не вводив дані кожного разу при роботі з додатком;
- повинна бути історія надання послуг з розрахунку академічної заборгованості (ПІБ студента, дисципліна, ПІБ викладача який викладав цю дисципліну, кількість годин, сума, дата формування документа з розрахунку академічної заборгованості);
- у додатку повинно бути 2 основні функції: розрахунок академічної заборгованості за предметом та розрахунок академічної заборгованості за послуги екзаменаційної комісії;
- для швидкості та спрощення процесу заповнення інформації необхідної для розрахунку академічної заборгованості потрібно використовувати API бази даних «КПІ ім. Ігоря Сікорського»;
- у разі відсутності доступу до мережі Internet повинен бути маркер, що вказує на це (не буде доступу до API);

- одразу після заповнення всієї інформації повинен відкриватися сформований документ у форматі Word, який складається з трьох частин;
- інформація про надані освітні послуги студенту записується в базу даних у вигляді таблиці Excel;
- сформований документ у форматі Word зберігається в обраній теці;

Після сформульованих критеріїв було проаналізовано сучасні популярні мови програмування та обрано мову написання програмного забезпечення (табл. 2.1).

Таблиця 2.1

Критерії вибору мови програмування для розроблення додатку

Мова програмування	Критерії			
	Кросплатформність	Встановлення додаткового ПЗ	Швидкість	Автоматична робота з пам’яттю
C++	+	-	+	-
C#	-	-	+-	+
Java	+	-	-	+
Python	+	-	+-	+

2.2 Обрані технології

Проаналізувавши техніко-програмні засоби, що використовуються в ВНЗ та мови які б могли задовільнити роботі обрано найефективнішу комбінацію «Залізо-ПЗ» (табл. 2.2).

Таблиця 2.2

Порівняння наявної технічної бази з мовами програмування-претендентами на вибір

	C++	C#	Java	Python
ОС Windows	+	+	+	+
Переваги	Висока швидкість, легкість вивчення, велика кількість стандартних рішень	Автоматичне виділення пам’яті, велика кількість стандартних рішень	Автоматичне виділення пам’яті, велика кількість стандартних рішень	Складність навчання
Недоліки	Ручне виділення пам’яті	Лише для ОС Windows	Тільки back-end, низька швидкість	Суперечності між версіями мови

Висновки до розділу

Було підібрано спочатку аналітичним, а потім дослідницьким шляхом оптимальні технології для створення ГРС.

Орієнтація на ОС Windows, тому що усі ВНЗ використовують саме цю ОС, підтримка з версії Windows 7, тому що це найбільш використовувана версія ОС.

Мова програмування C#, тому що ця мова програмування була спеціально розроблена під ОС Windows. Фреймворк .NET (підтримка з версії 4.0.0. до останньої на сьогоднішній день версії 4.7.2.), тому що версія .NET 4.0.0. підтримується починаючи з Windows 7.

Використання API БД «КПІ ім. Ігоря Сікорського» для пришвидшення роботи з додатком шляхом підказок з вибору студента, викладача, дисциплін.

Функціонал додатку дозволить проводити усю обчислювальну частину формуючи документи з НОП або ЕК. У подальшому, завдяки стратегії описаній у розділі «Розроблення стартап-проектів», практичну частину

розроблену у рамках магістерської дисертації можна буде використовувати для всіх ВНЗ України, а згодом, її можна інтегрувати до повноцінної системи електронного врядування для усіх освітніх закладів, що переведе галузь освіти на новий якісніший рівень.

Високоякісний програмний продукт повинен передбачати усі можливі відхилення від «сценарію», тому у даній магістерській дисертації підібрана оптимальна комбінація рішень, що зменшує вірогідність будь-яких відхилень. Сформульовано та обґрунтовано вибір усіх технологій, які використовуватимуться в подальшому.

В даному розділі було описано вимоги до додатку та архітектура всієї системи, та окремих рівнів, базуючись на цих вимогах. Також були обрані технології для розробки. На основі обраної архітектури, функціональних вимог було розроблено сервіс для розрахунку академічної заборгованості з надання освітніх послуг. Отже, можемо перейти до тестування системи.

РОЗДІЛ 3. ПРАКТИЧНЕ ЗАСТОСУВАННЯ

3.1. Структура гнучкого роботизованого сервісу

Розрахунок академічної заборгованості заключається в тому, що визначається які предмети були надані студенту (суттєву роль відіграє навчальний план, тому що від типу заліку та робіт, які повинні бути виконані в ході залежить вартість), яким викладачем (від наукового ступеня залежить вартість послуг) були надані освітні послуги, курсу студента (для різних курсів зазначені різні нормативні години для складання дисциплін) та кількість годин відведених на певний предмет (рис.3.1).

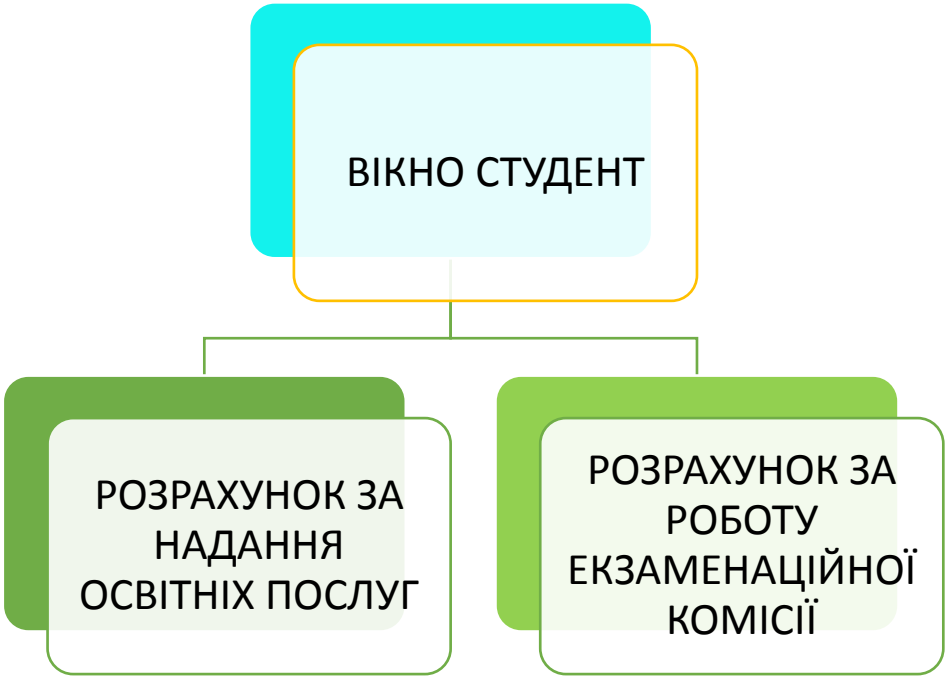


Рисунок 3.1. Структура роботи сервісу

У вікні «Студент» обираються наступні параметри:

- ПІБ студента;
- Курс студента (1-2 або 3-6);

У вікні «Розрахунок за надання освітніх послуг» обираються наступні параметри:

- ПІБ викладача;
- Назва дисципліни;
- Вчене звання викладача;

- Вид занять;
- Тип зарахування;

3.2 Інформаційні структури даних

3.2.1 Версія 1.0.0.

1. Розробка вікна «Студент»

- Поля: Прізвище, ім’я, по батькові (рис. 3.2)

Рисунок 3.2. Приклад вводу ПІБ студента у першій версії ГРС

- Вибір курсу за допомогою кнопок Radio Button: I-II або III-VI (рис. 3.3)

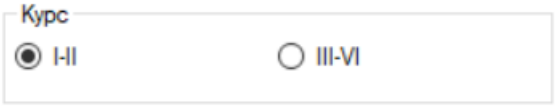


Рисунок 3.3. Приклад вибору курсу за допомогою кнопок Radio Button у першій версії ГРС

2. Розробка вікна «Налаштування»

- Область «Ціна в залежності від наукового звання» (асистент; старший викладач; кандидат технічних наук; доцент, кандидат технічних наук; доктор технічних наук; професор, доктор технічних наук) (рис.3.4);

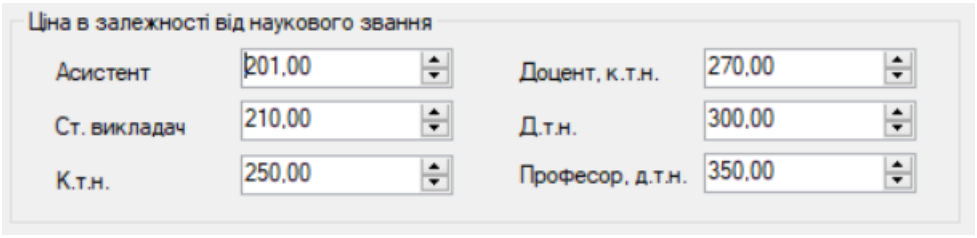


Рисунок 3.4. Приклад області «Ціна в залежності від наукового звання» у першій версії ГРС

- Область «Норми часу для I-II курсу» (лабораторні, практичні, курсові проекти, контрольні роботи, розрахунково-графічні роботи, реферати, залік, екзамен) (рис. 3.5);

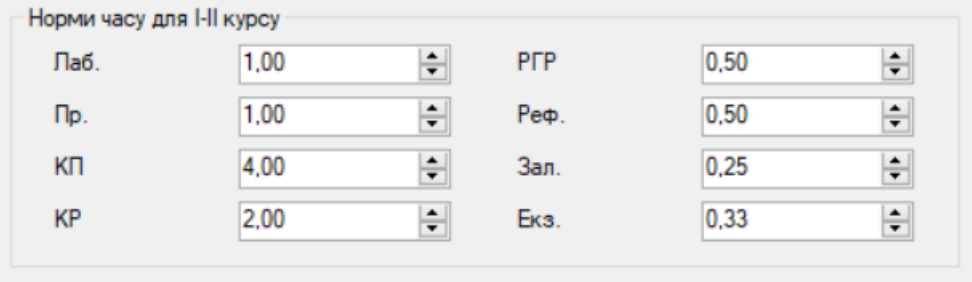


Рисунок 3.5. Приклад області «Норми часу для I-II курсу» у першій версії ГРС

- Область «Норми часу для III-VI курсу» (лабораторні, практичні, курсові проекти, контрольні роботи, розрахунково-графічні роботи, реферати, залік, екзамен) (рис. 3.6);

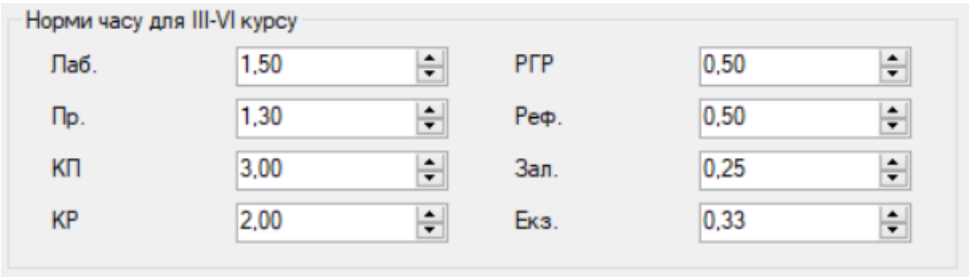


Рисунок 3.6. Приклад області «Норми часу для III-VI курсу» у першій версії ГРС

- Скасування, очищення, збереження введених параметрів (рис. 3.7);

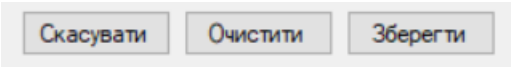


Рисунок 3.7. Кнопки дій у вікні «Налаштування»

3. Перехід у вікно «НОП» (рис. 3.8)

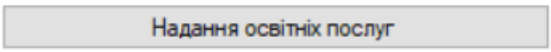


Рисунок 3.8. Кнопка переходу у вікно «НОП»

4. Розробка вікна «НОП» (рис. 3.9)

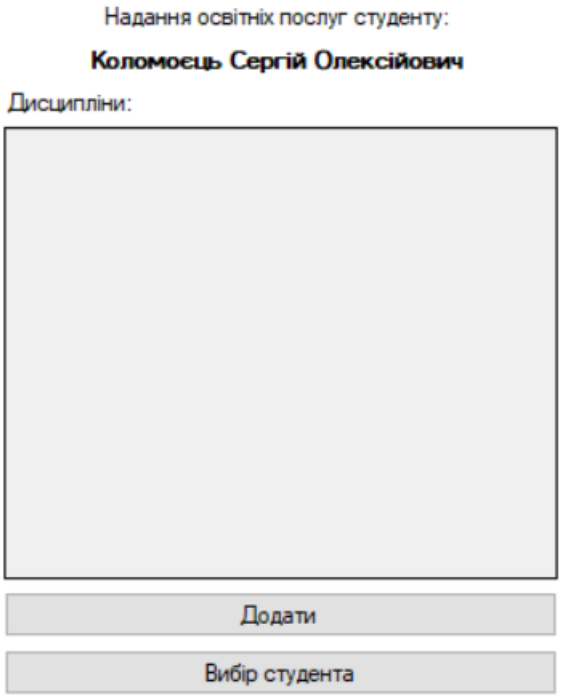


Рисунок 3.9. Незаповнене вікно «НОП» для обраного студента

5. Розробка вікна «Викладач»

- Поля: Прізвище, ім’я, по батькові (рис. 3.10);

Рисунок 3.10. Приклад полей ПІБ викладача у першій версії ГРС

- Поле «Назва дисципліни» (рис. 3.11);

Рисунок 3.11. Приклад поля «Назва дисципліни» у першій версії ГРС

- Область вибору «Вчене звання викладача» (рис. 3.12);

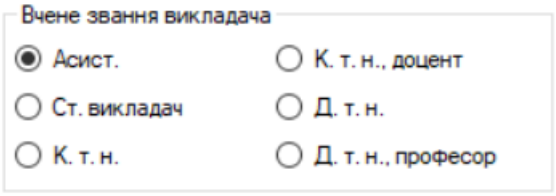


Рисунок 3.12. Приклад області вибору «Вчене звання викладача» у першій версії ГРС

- Область вибору «Вид занять» (рис. 3.13);

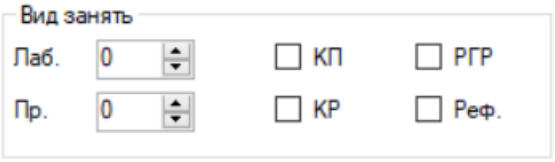


Рисунок 3.13. Приклад області вибору «Вид занять» у першій версії ГРС

- Область вибору «Тип зарахування» (рис. 3.14);

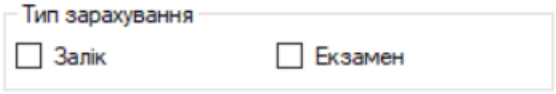


Рисунок 3.14. Приклад області вибору «Тип зарахування» у першій версії ГРС

- Кнопки дій у вікні «Викладач» (рис. 3.15);

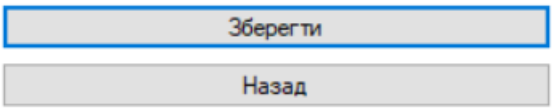


Рисунок 3.15. Приклад кнопок дій у вікні «Викладач» у першій версії ГРС

6. Генерація документації з НОП

3.2.2 Версія 1.0.1.

1) Шлях для збереження файлів (рис. 3.16);

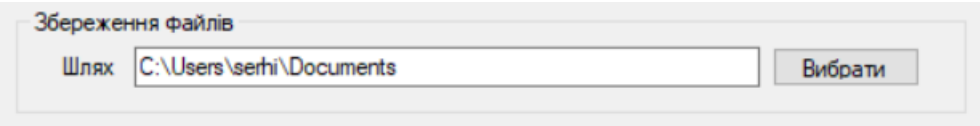


Рисунок 3.16. Шлях для збереження файлів

3.2.3 Версія 1.0.2.

1) Історія надання студентам ОП (записується у автоматично згенеровану таблицю Excel після першої генерації документів) (рис. 3.17, рис. 3.18);

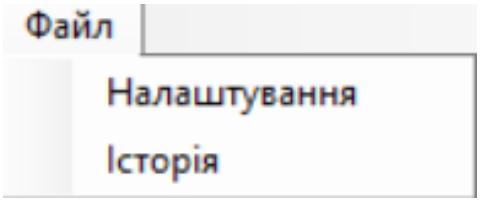


Рисунок 3.17. Кнопка «Історія НОП»

	A	B	C	D	E
1	ПІБ студента	Група	Дисципліна	Викладач	
2	Коломоєць Сергій Олексійович	ІК-82мп	Дискретна математика	Ліхоузова Тетяна Анатоліївна	
3	Коломоєць Сергій Олексійович	ІК-82мп	Гнучкі комп'ютеризовані системи - 2. Проектування ГКС	Ткач Михайло Мартинович	
4	Коломоєць Сергій Олексійович	ІК-82мп	Системне програмування	Лісовиченко Олег Іванович	
5					
6					
7					

Рисунок 3.18. Історія НОП у таблиці Excel

3.2.4 Версія 2.0.0.

1. Імплементація API (рис. 3.19);

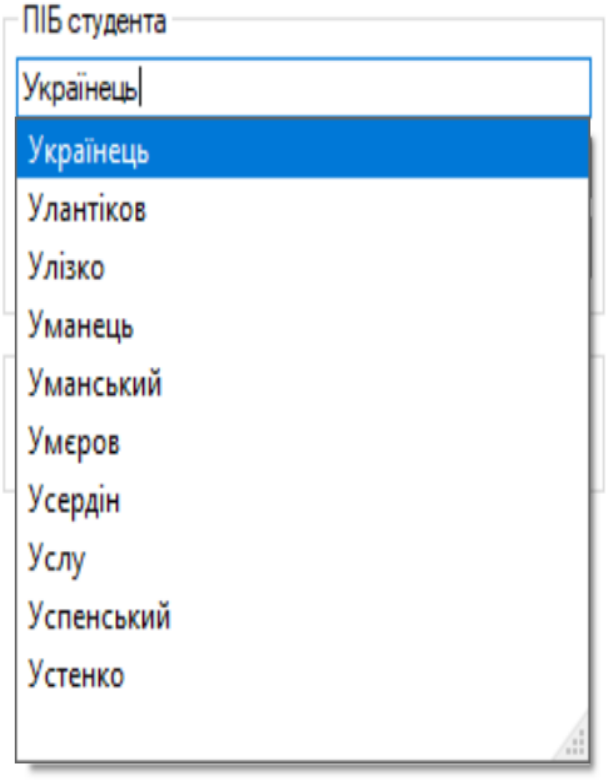


Рисунок 3.19. Робота API

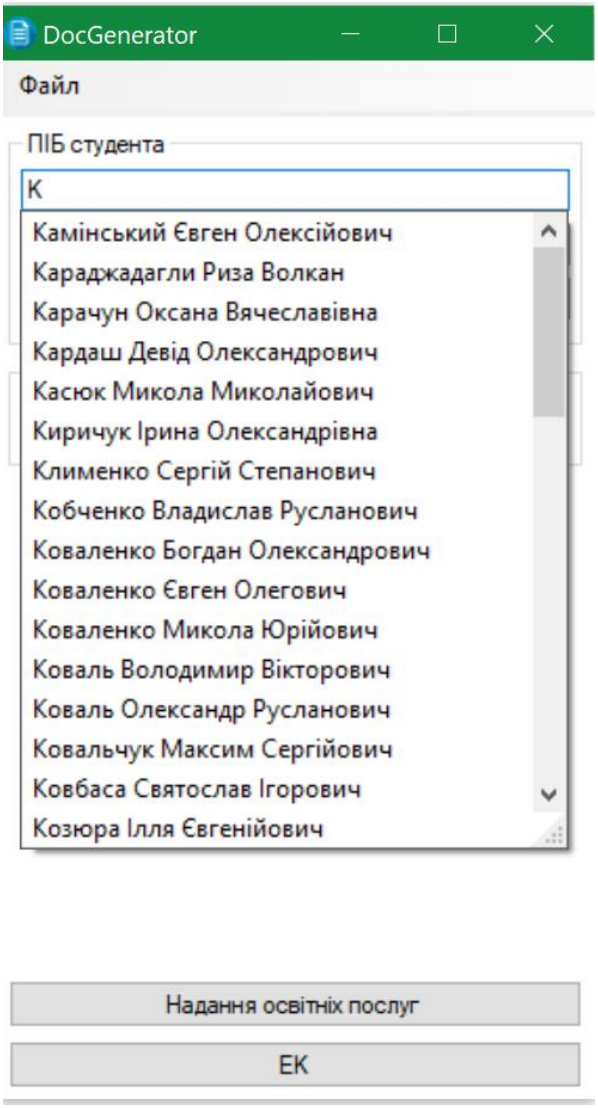


Рисунок 3.20. Робота API оновленого варіанту

2. Перевірка роботи API на наявність студента (рис. 3.21);

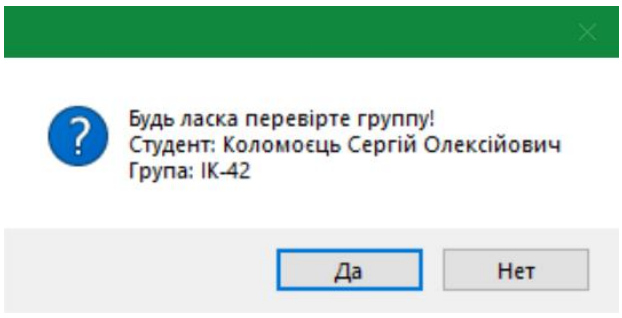


Рисунок 3.21. Перевірка роботи API на коректність вибору студента

3.2.5 Версія 2.0.1.

1. Розробка вікна «ЕК» (рис. 3.22).

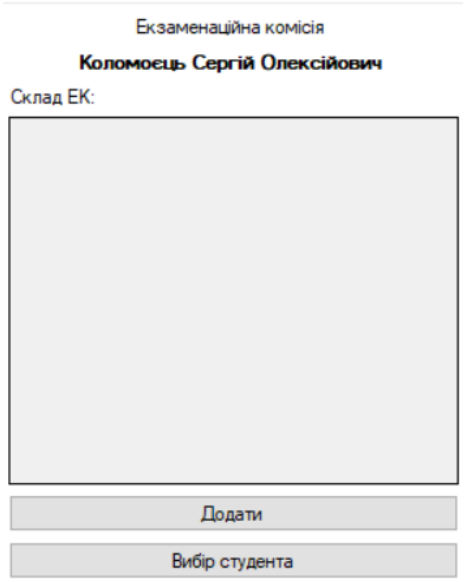


Рисунок 3.22. Початковий стан вікна «ЕК» для обраного студента

2. Розробка вікна «Викладач 2»

- Поля: Прізвище, ім’я, по батькові (рис. 3.23);

Рисунок 3.23. Приклад полей ПІБ викладача 2

- Область вибору «Вчене звання викладача» (рис. 3.24);

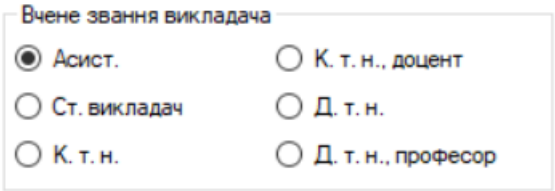


Рисунок 3.24. Приклад області вибору «Вчене звання викладача 2»

- Область вибору «Вид занять» (рис. 3.25).

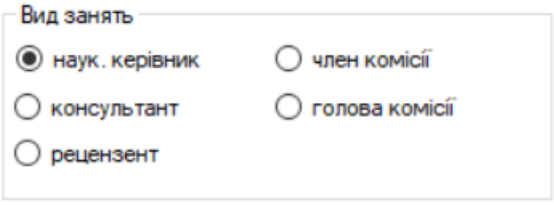


Рисунок 3.25. Приклад області вибору «Вид занять»

- Область вибору «Кількість годин» (рис. 3.26);

Рисунок 3.26. Приклад області вибору «Кількість годин»

- Кнопки дій у вікні «Викладач» (рис. 3.27);

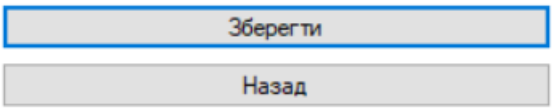


Рисунок 3.27. Приклад кнопок дій у вікні «Викладач 2»

3.3 Опис функцій

Одразу після запуску сервісу відкривається початкове вікно «Студент» (рис. 3.28). При вводі перших літер прізвища з’являється список з запропонованих студентів, які є в БД «КПІ ім. Ігоря Сікорського» (рис. 3.28).

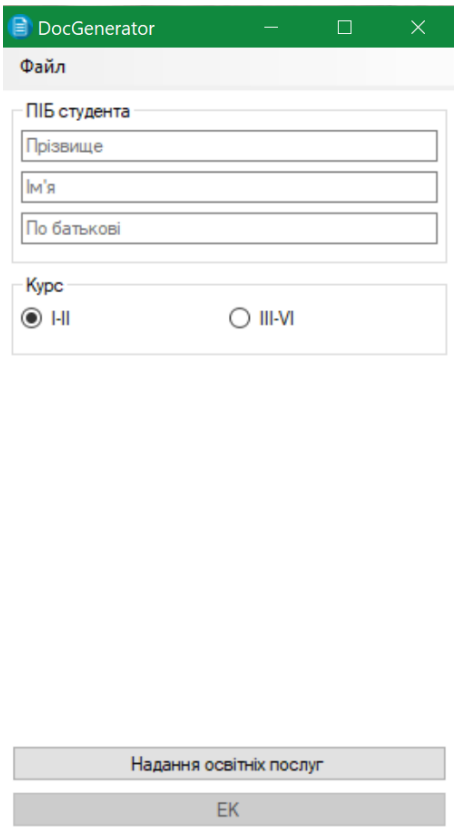


Рисунок 3.28. Початкове вікно розробленого ГРС

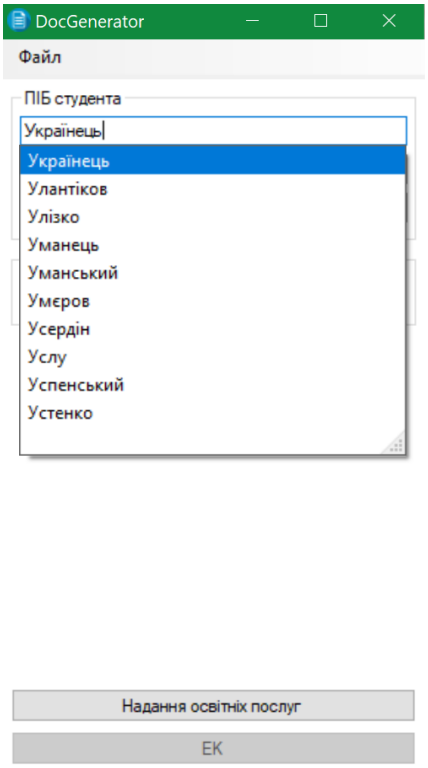


Рисунок 3.29. Робота API у розробленому ГРС

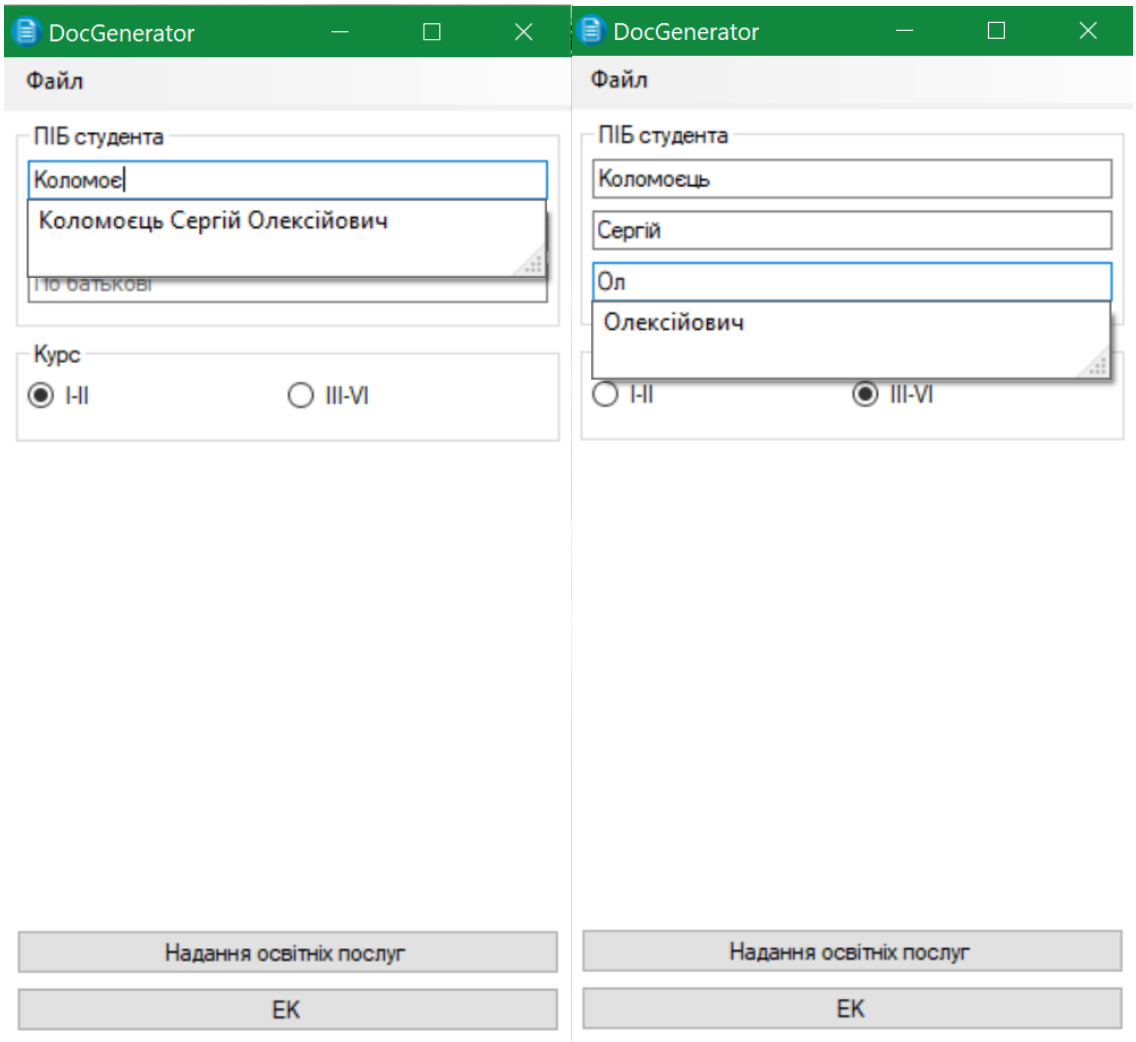


Рисунок 3.30. Підказки при виборі ПІБ студента

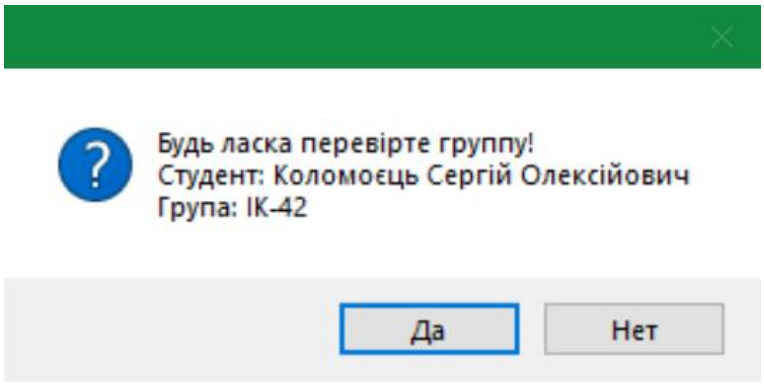


Рисунок 3.31. Перевірка обраних даних на наявність інформації про студента у БД

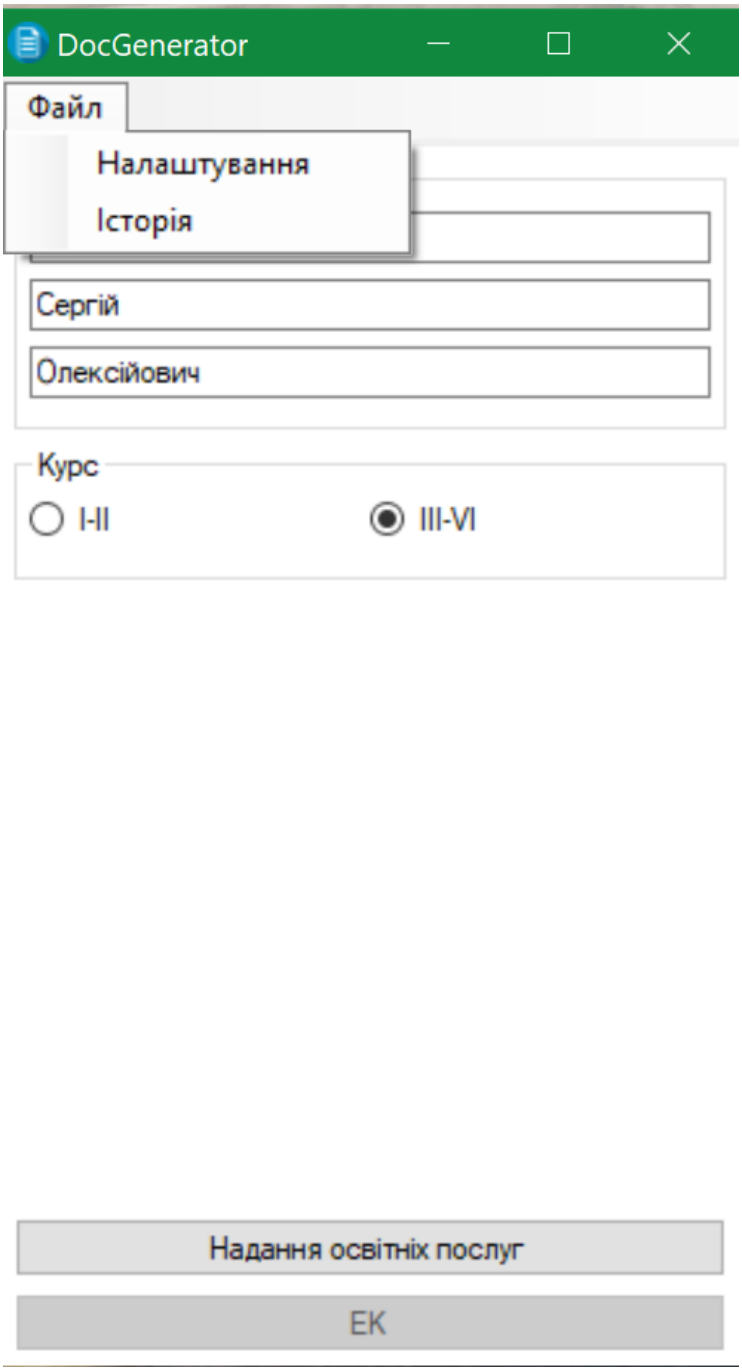


Рисунок 3.32. Перехід до меню «Налаштування»

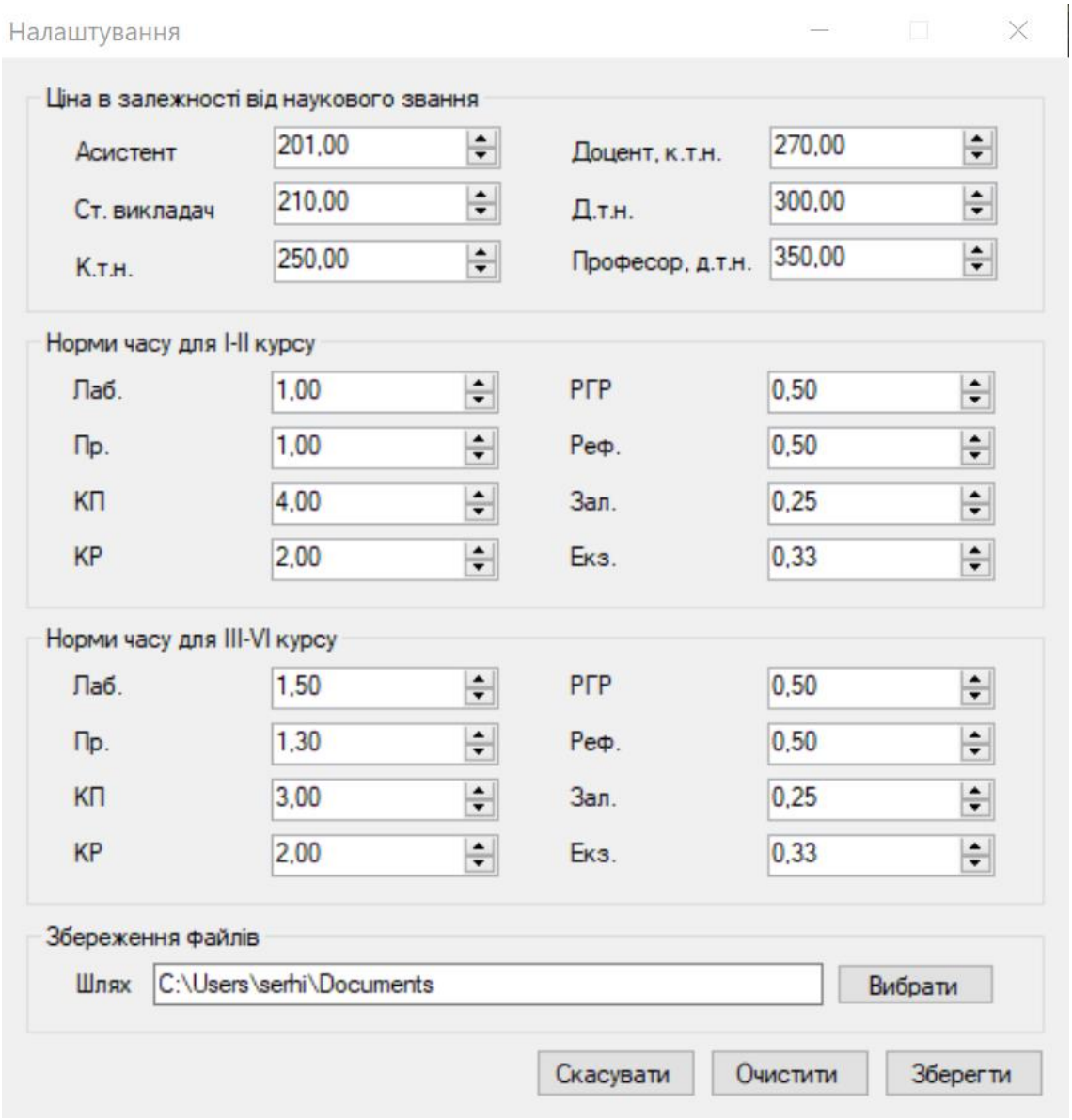


Рисунок 3.33. Меню «Налаштування»

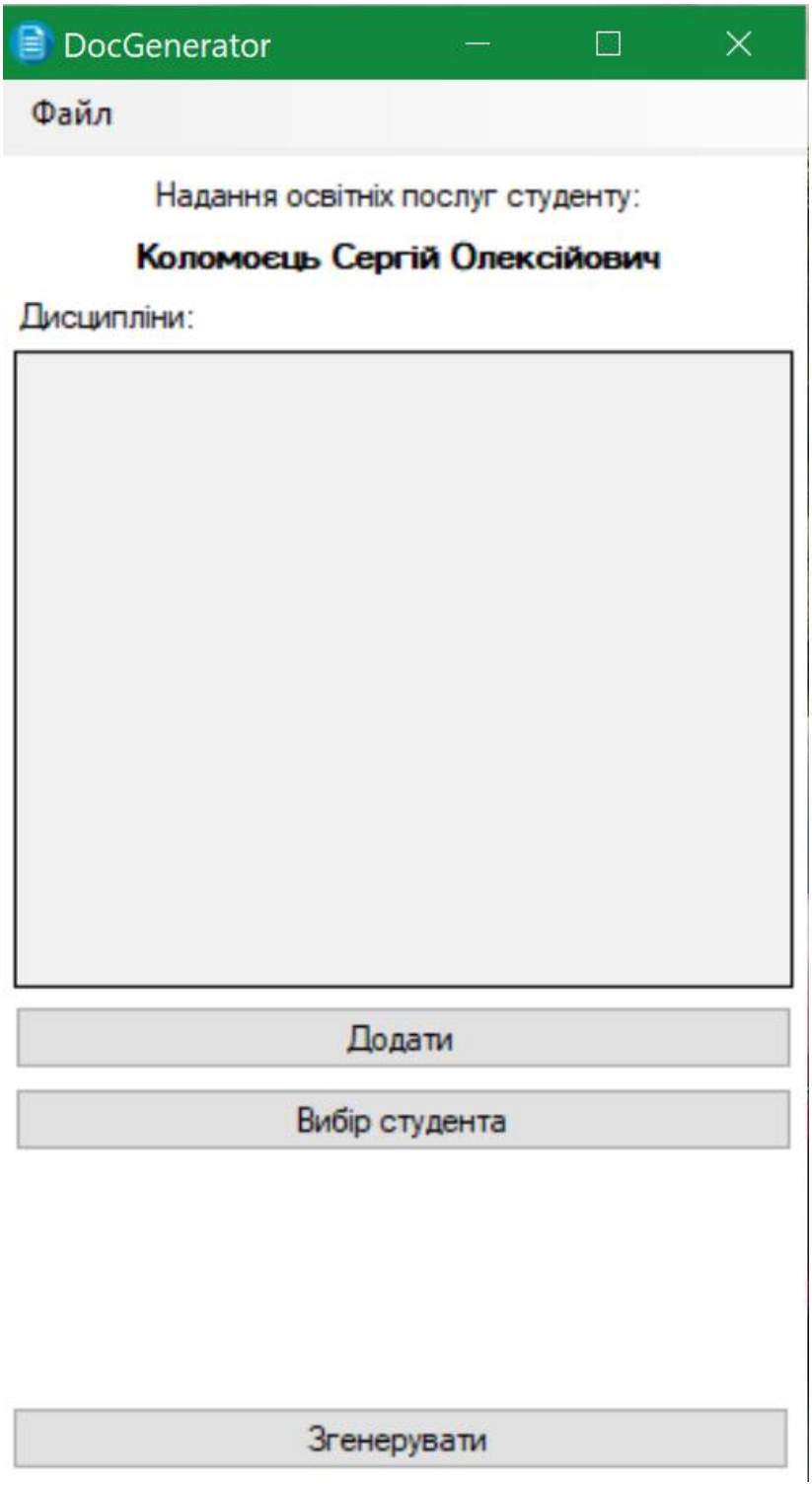


Рисунок 3.34. Початкове вікно меню «НОП»

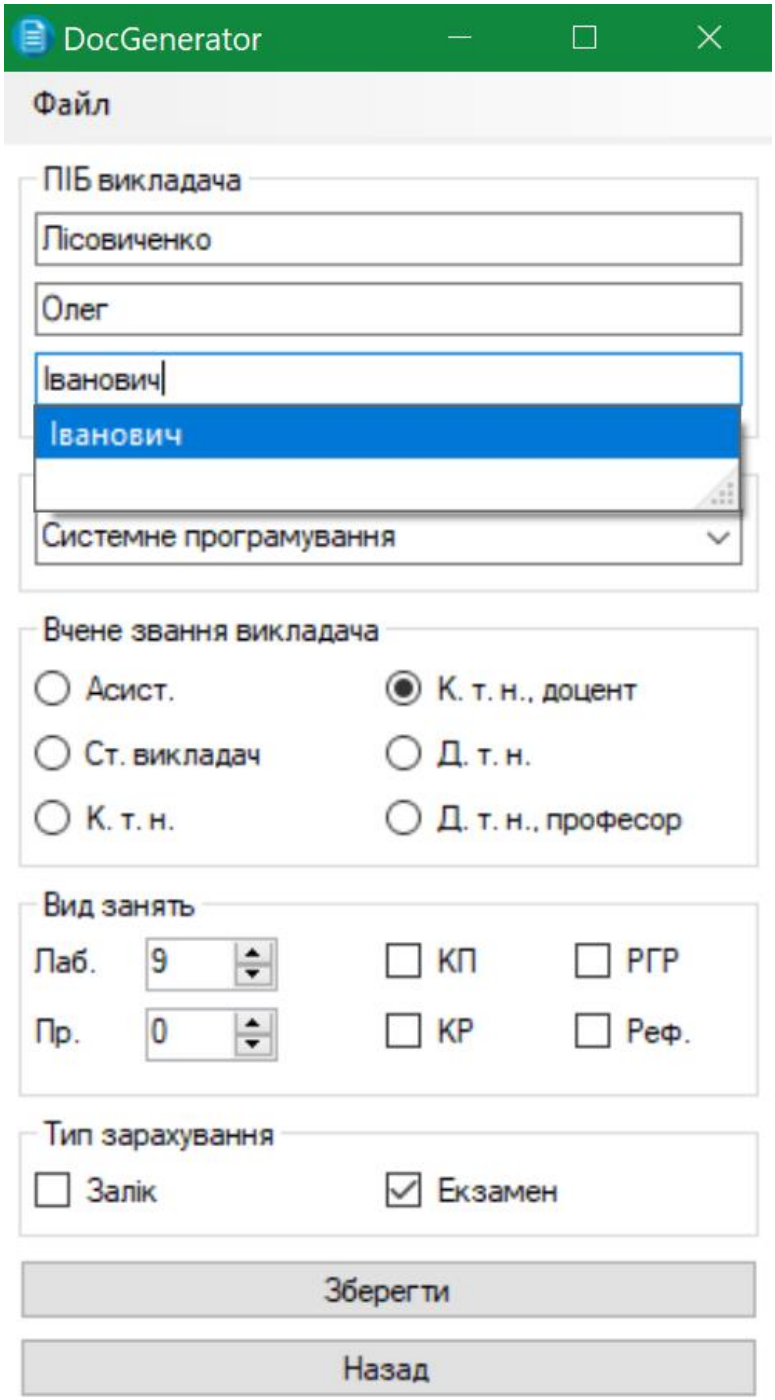


Рисунок 3.35. Використання API для допомоги у виборі викладача у вікні «Викладач»

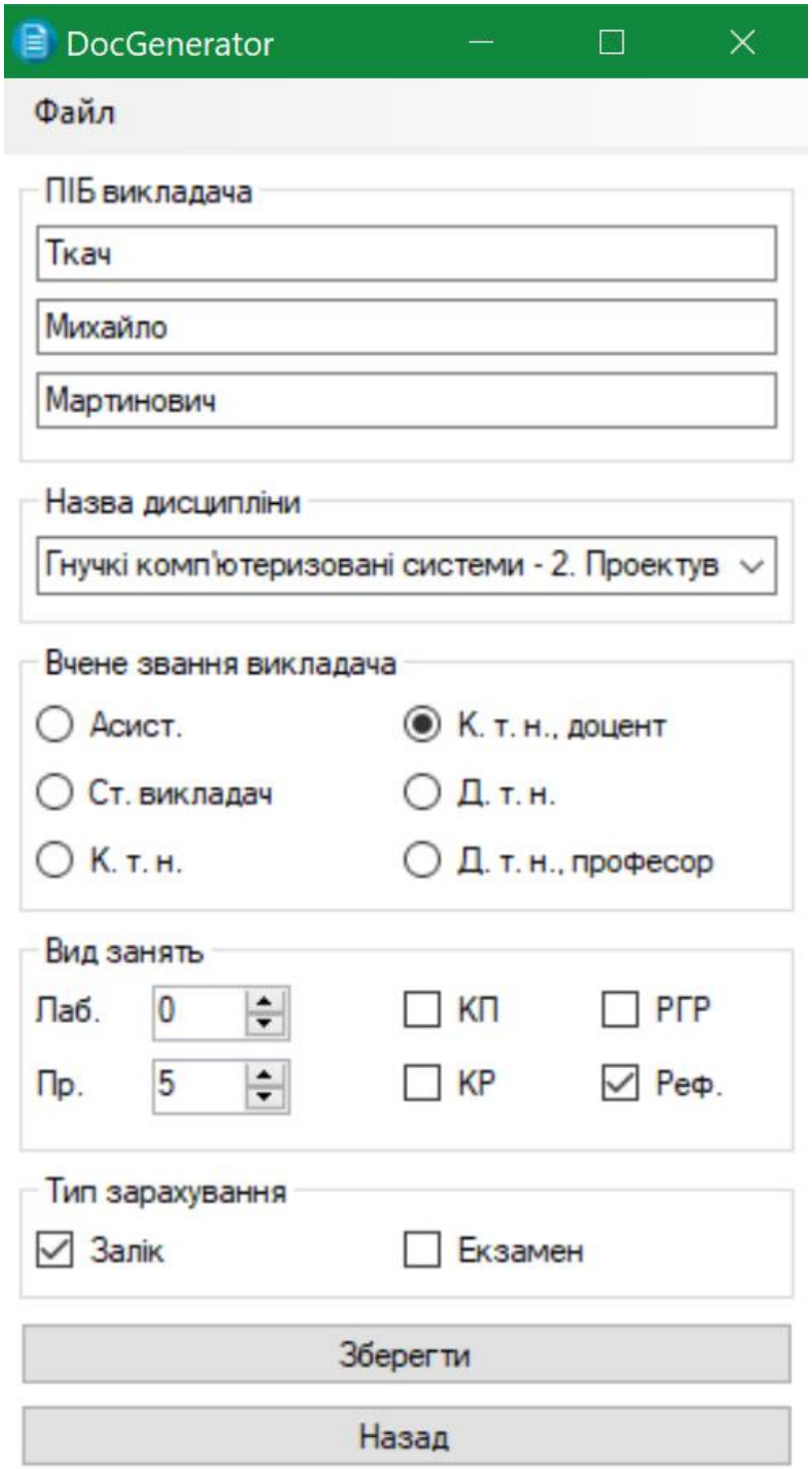


Рисунок 3.36. Заповнене вікно «Викладач»

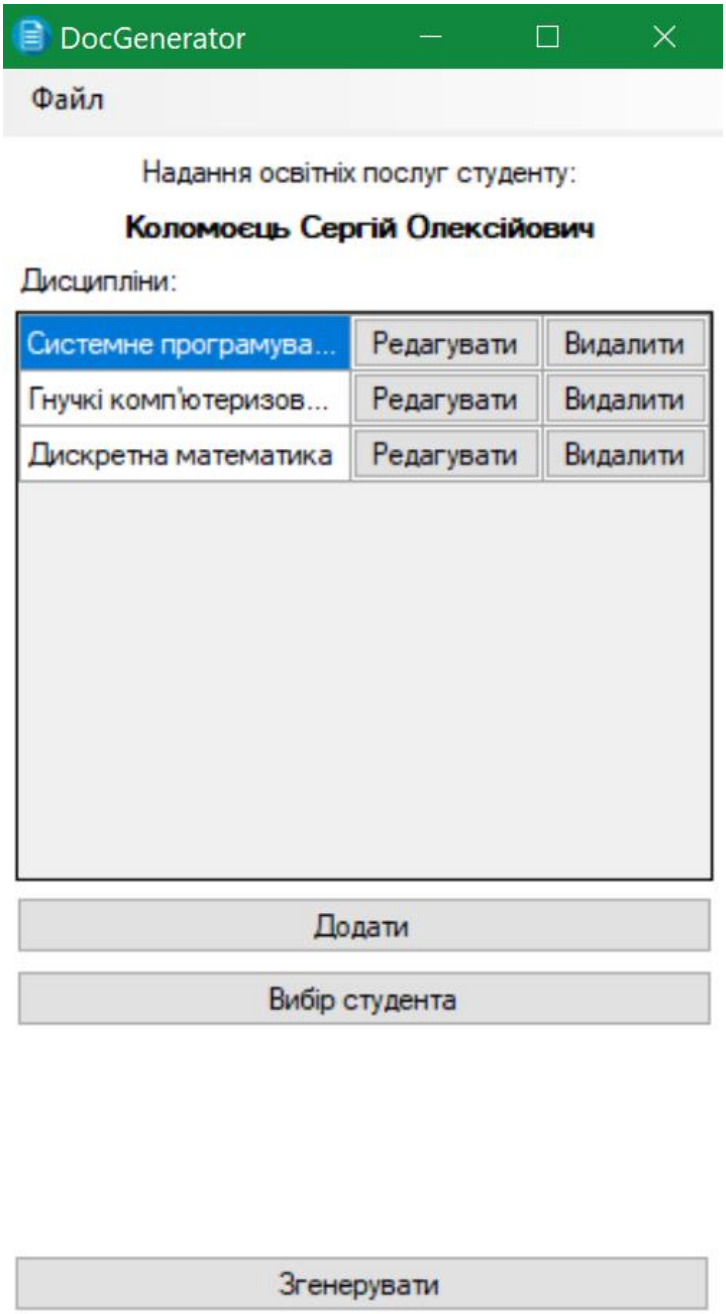


Рисунок 3.37. Заповнене вікно «НОП»

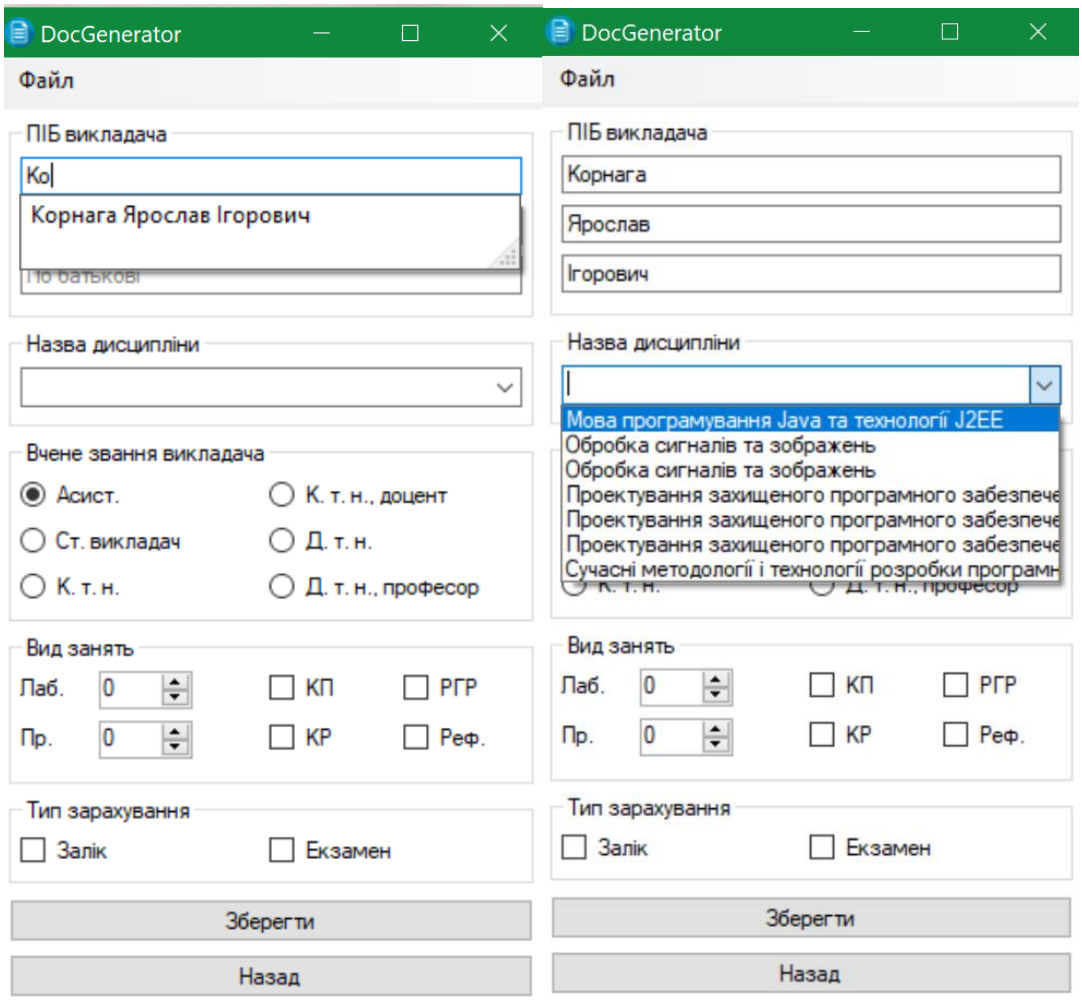


Рисунок 3.38. Вибір дисципліни, яка викладається

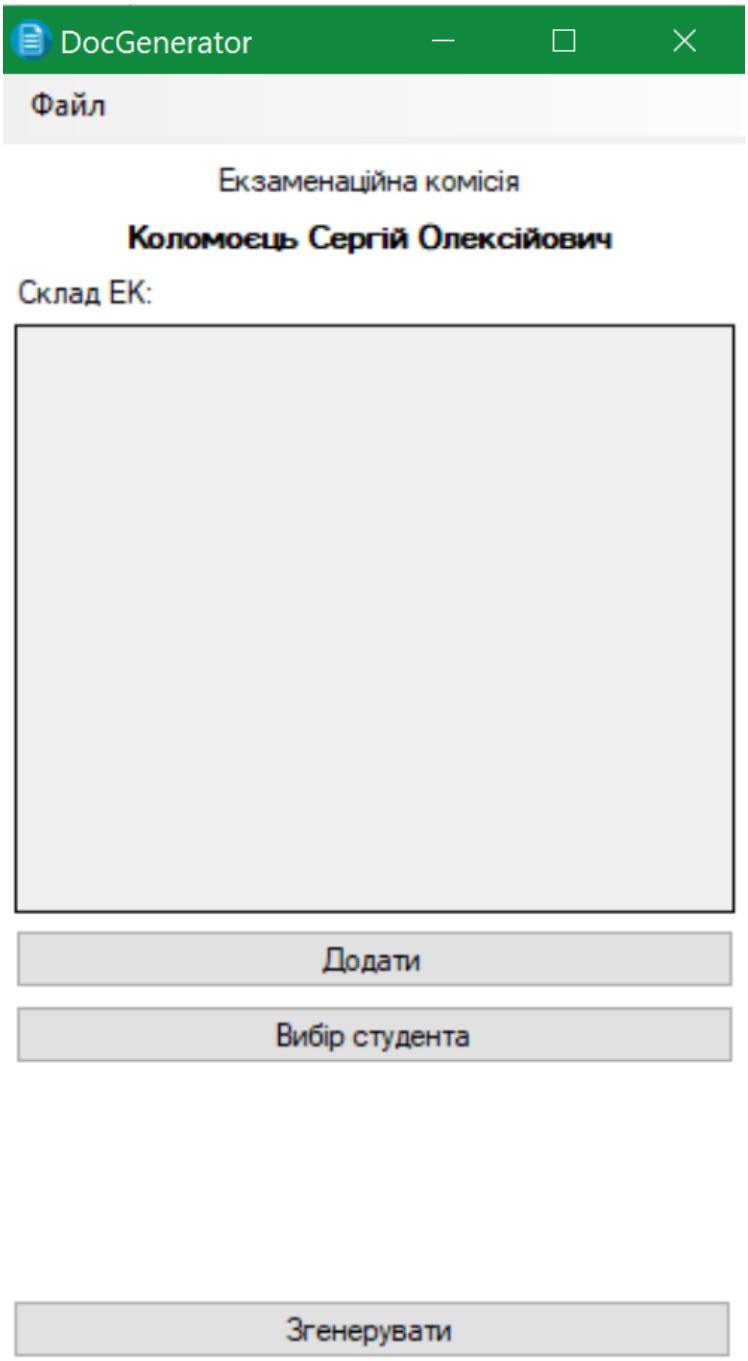


Рисунок 3.39. Початкове вікно «Склад ЕК»

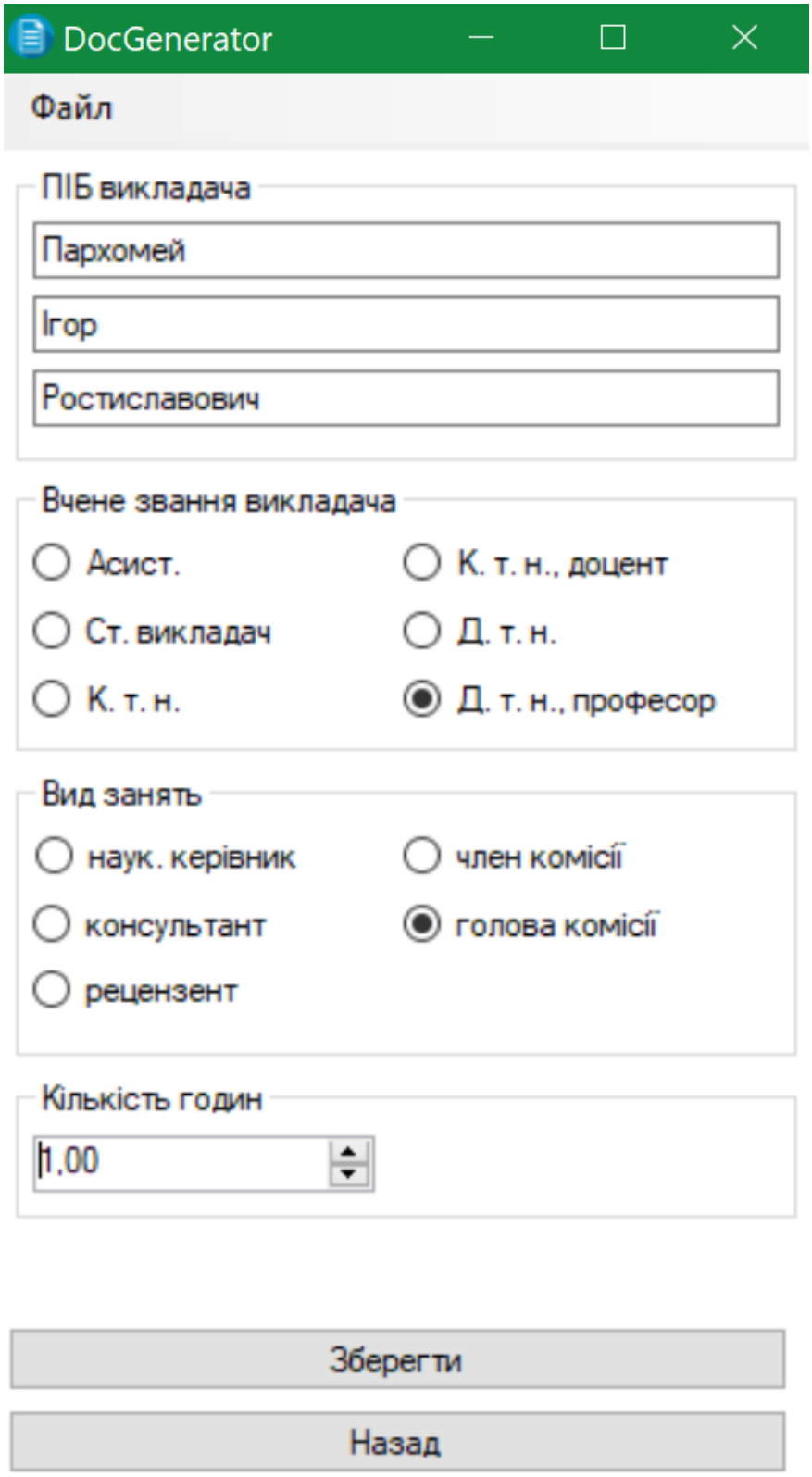


Рисунок 3.40. Приклад заповнення вікна «Викладач 2» для голови комісії

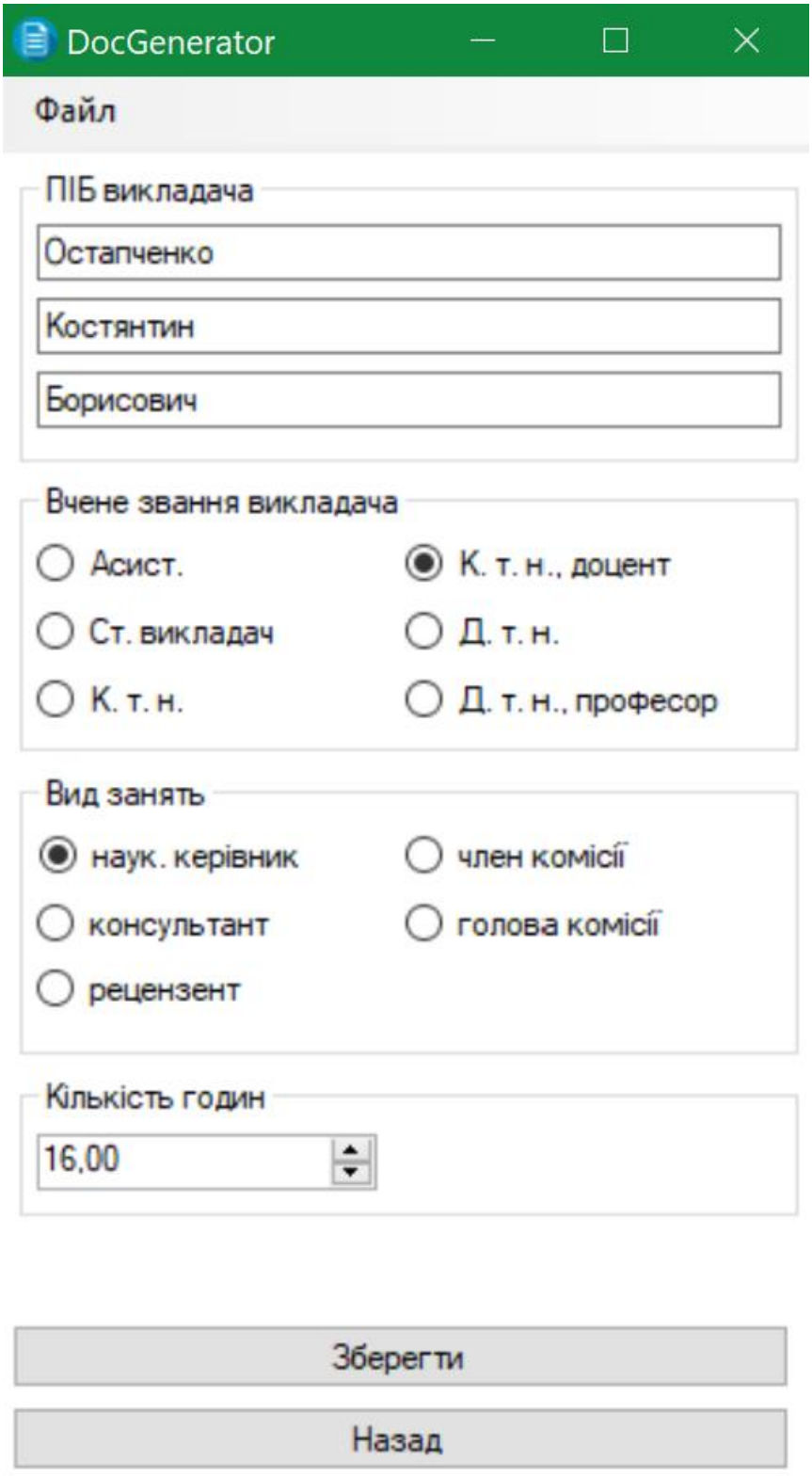


Рисунок 3.41. Приклад заповнення вікна «Викладач 2» для наукового керівника

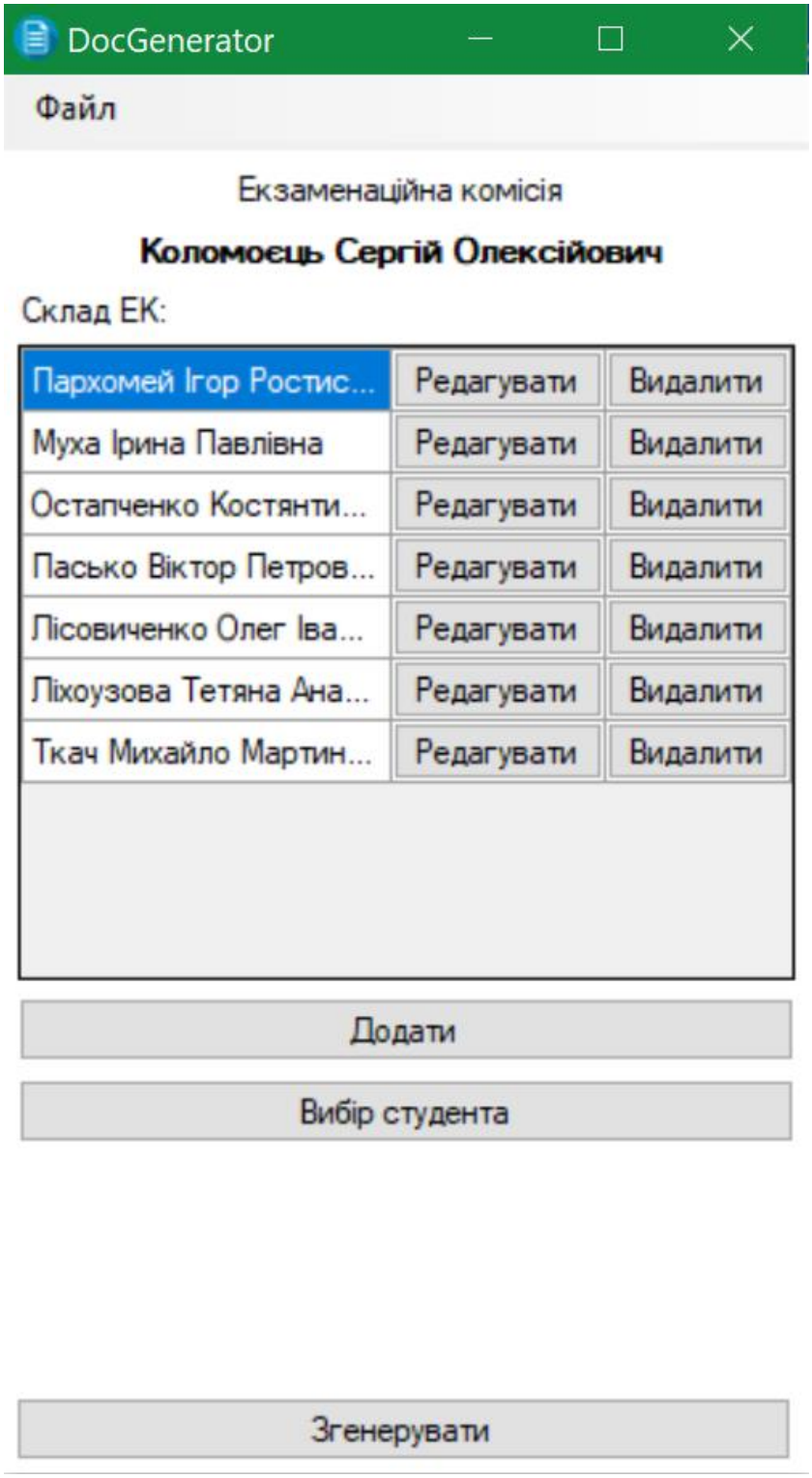


Рисунок 3.42. Заповнене вікно «Склад ЕК»

Додаток № _____ до договору № _____
від «____» _____ 20___ р.

Р О З Р А Х У Н О К
освітніх послуг

Коломоєць Сергій Олексійович
(прізвище, ім'я, по батькові)

Системне програмування
(назва дисципліни)

Вид занять	Лаб	Пр.	КП	КР	РГР	Реф	Зал	Екз
К-сть годин, год. (є, немає)	13,5	-	~~Є~~ Н	~~Є~~ Н	~~Є~~ Н	~~Є~~ Н	~~Є~~ Н	Є ~~Н~~
Норми часу год/год.(год) по виду занять	1,5	1,3	3	2	0,5	0,5	0,25	0,33
Вартість години освітніх послуг, грн/год	270	-	-	-	-	-	-	270
Вартість по виду занять, грн.	3645	-	-	-	-	-	-	89,1

Сума: 3734,1 грн. (13,83 год.)

Загальна вартість: 3734,1 грн. (13,83 год.)

Замовник _____ Виконавець _____

Рисунок 3.43. Приклад згенерованого документа «Розрахунок освітніх послуг»

Акт виконаних робіт № _____

«_____» _____

Ми, що нижче підписалися, Факультет інформатики та обчислювальної техніки (ФІОТ) Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», в особі декана Теленика С.Ф., надалі Виконавець та Коломоєць Сергій Олексійович, надалі Замовник, склали цей акт про те, що Виконавцем були надані освітні послуги за наступними дисциплінами:

Системне програмування	3734,1 грн. (13,83 год.)

Загальна вартість освітніх послуг складає 3734,1 грн. , без ПДВ. Послуги надані в строк і в повному обсязі.

Замовник Виконавець

Рисунок 3.44. Приклад згенерованого документа «Акт виконаних робіт»

Висновки до розділу

Реалізовано сервіс з розрахунку академічної заборгованості з надання освітніх послуг. На основі результатів роботи програми було створене керівництво для користувачів, які мають ролі «Розробник» і «Користувач». В керівництво було включено опис таких операцій, як введення даних, та його редагування та навігації по вікнам меню. Було описано 2 основних принципи роботи сервісу: «Розрахунок надання освітніх послуг» та «Розрахунок за роботу екзаменаційної комісії».

Отже, ми можемо переходити до маркетингово аналізу стартап-проекту.

РОЗДІЛ 4. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП ПРОЕКТУ

Проведемо маркетинговий аналіз стартап-проекту задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

4.1 Опис ідеї проекту

Розглянемо зміст ідеї, можливі напрямки застосування та основні вигоди, які може отримувати користувач від проекту.

Таблиця 4.1

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробити сервіс для розрахунку академічної заборгованості з надання освітніх послуг	Розрахунок НОП	Можливість безпомилкового розрахунку НОП конкретному студенту, економія часу за рахунок автоматизації процесу, зниження ймовірності помилки вибору студента/викладача/предмету через використання API БД, що надає підказки при виборі вищезазначених пунктів
	Розрахунок за послуги ЕК	Можливість безпомилкового розрахунку за послуги ЕК конкретному студенту, економія часу за рахунок автоматизації процесу, зниження ймовірності помилки вибору студента/викладача/предмету через використання API БД, що надає підказки при виборі вищезазначених пунктів

Тепер визначимо які техніко-економічні переваги має даний додаток над іншими існуючими рішеннями. Як було виявлено при аналізі найближчих

конкурентів, у даного сервісу не існує конкурентів, що зробить його монополістом при виході на ринок розрахунку академічної заборгованості та електронного урядування в ВНЗ.

Таблиця 4.2

Визначення сильних, слабких, нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів		W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Конкурент 1			
1	Введення тексту власноруч	+	-			Сервіс буде функціонувати навіть за відсутності доступу до мережі Internet
2	Автоматизоване введення тексту	+	-			Швидкість роботи з сервісом
3	Наявність БД	+	-	Працює за наявності доступу до мережі Internet		Швидкість роботи з сервісом
4	Автоматична генерація документів	+	-			Відмова від використання людської праці
5	Розрахунок вартості НОП/ЕК	+	-			

Продовження таблиці 4.2

6	Розрахунок часу витраченого на НОП/ЕК	+	-			
7	Можливість адаптувати вихідний документ та БД для поточного користування	+	-			Сервіс буде мультифункціональним і не буде прив’язаний до одного документа

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності.

4.2 Технологічний аудит ідеї проекту

Проведемо аудит технологій, за допомогою яких можна реалізувати ідею проекту та визначимо технологічну здійсненність ідеї проекту.

Таблиця 4.3

Технології здійснення ідеї проекту

№	Ідея проекту	Технології реалізації	Наявність технологій	Доступність технологій
1	Автоматизація формування документообігу	Розроблюваний додаток	-	-
2	Прикладний рівень	Мова програмування C#, фреймворк .NET	+	+

Продовження таблиці 4.3

3	Рівень даних	Використання API БД «КПІ ім. Ігоря Сікорського»	+	+
4	Допоміжні програмні засоби	MS Word (для формування документа), MS Excel (для збереження бази даних наданих ОП)	+	+
Обрана технологія реалізації проекту: розроблюється система автоматизованого формування документів для розрахунку академічної заборгованості студента з НОП/ЕК з використання БД та стандартних програмних засобів MS Office.				

Проаналізувавши вищенаведену інформацію можна зробити висновок, що при наявності деяких програмних засобів, як середа розробки для мови програмування C# — Visual Studio та стандартних програм для редагування документації Microsoft Office (Word, Excel) технологічна реалізація даного сервісу можлива. Для роботи з цим сервісом користувач повинен мати доступ до мережі Internet (необов'язково, але пришвидшує роботу) та стандартні офісні програми Microsoft Office (Word, Excel). Так як реалізація продукту можлива, перейдемо до аналізу ринкових можливостей запуску стартап-проекту.

4.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначимо ринкові можливості, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту.

Таблиця 4.4

Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	0
2	Загальний обсяг продажу, грн./ум.од	0
3	Динаміка ринку	Потрібен розвиток
4	Наявність обмежень для входу	Наявність програм MS Office (Word, Excel)
5	Специфічні вимоги для стандартизації та специфікації	-
6	Середня норма рентабельності в галузі/ринку, %	0

Зважаючи на повністю відкритий ринок можливостей та відсутність конкурентів ринок може бути монополізованим першим надавачем подібних послуг та навіть при появі конкурентів втримати більш ніж 50% загального обсягу ринку.

Таблиця 4.5

Характеристика потенційних клієнтів стартап-проекту

№	Потреба що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Відсутність автоматизованих програмних засобів для розрахунку АЗ	ВНЗ	-	Можливість швидкого формування документації для НОП
2		Приватні надавачі освітніх послуг (курси, репетитори)		

Після визначення потенційних груп клієнтів проведемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому

впровадженню проекту, та факторів, що йому перешкоджають. Фактори в табл. 4.6, 4.7 подаються в порядку зменшення значущості.

Таблиця 4.6

Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Конкуренти	Наявність конкурентів котрі надають схожі рішення	Розробка унікальних характеристик товару; Подальша підтримка системи
2	Кошти на розробку та підтримку продукту	Закінчення грошей та недостатнє фінансування	Пошук альтернативних джерел фінансування, таких як реклама в додатку, нових інвесторів, пошук дешевших рішень для розгортки системи та зберігання даних
3	Відсутність необхідних умов для використання сервісу	Відмова від продукту	Надання додаткових програмних засобів MS Office (Word, Excel) разом з розробленим сервісом

Таблиця 4.7

Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Новий продукт	Надання нових рішень у сфері	Розробка унікальних характеристик товару
2	Використання БД	Швидкість роботи	Створення розробником локальної бази з можливістю редагування

Продовження таблиці 4.7

3	Низька вартість	Сервіс буде більш привабливим для користувачів через малу вартість по підписці	Пошук альтернативних джерел фінансування, таких як реклама в додатку, нових інвесторів, пошук дешевших рішень для розгортки системи та зберігання даних для можливості надалі зберігати продукт безкоштовним

Далі наведено результат проведеного аналіз пропозиції де визначаються загальні риси конкуренції на ринку.

Таблиця 4.8

Ступеневий аналіз конкуренції та ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
1. Тип конкуренції: Монополія	Відсутність конкурентів	Оновлення програмного забезпечення та БД
2. Рівень конкурентної боротьби: Національний	Рішення можуть використовувати в начальних закладах на всій території держави	-
3. Галузева ознака: Внутрішньогалузева	Конкуренція в сфері освіти	Розробка нового функціоналу, контроль коректності даних
4. За видом товару: Товарно-родова	Різні функції формування документації	Вдосконалення графічного інтерфейсу, алгоритмів перекладу

Продовження таблиці4.8

5. За характером конкурентних переваг: Нецінова	Низька ціна, відсутність конкурентів та відсутність аналогів	Постійний розвиток та оновлення, врахування побажань замовників
6. За інтенсивністю: Не марочна	Немає спільної аудиторії через відсутність конкурентів	Інформування ринку щодо якості використовуваної новаторської системи, розширення функціоналу

Далі наведено огляд аналізу конкуренції, де проводиться більш детальний аналіз умов конкуренції в галузі

Таблиця 4.9

Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	-	-	-	Клієнти Можуть бути невдоволенні наявною функціональністю	Товари, які б задовольняли усі вимоги практично відсутні
Висновки	-	-	-	Вносити корективи згідно установ	-

Таблиця 4.10

Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Цінова політика	Отримання прибутку здійснюється в основному за рахунок реклами, що дає змогу зробити використання додатку безкоштовним

Продовження таблиці 4.10

2	Точність перекладу	Моніторинг варіантів перекладу із сторони спеціалістів забезпечує більшу його точність
3	Інтуїтивно зрозумілий інтерфейс	Користувачам легше взаємодіяти із системою
4	Платформи для навчання	Значне розширення функціоналу системи розрахунок того, що вона є підсистемою для навчальної платформи

За визначеними факторами конкурентоспроможності проведемо аналіз сильних та слабких сторін стартап-проекту.

Таблиця 4.11

Порівняльний аналіз сильних та слабких сторін проекту

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні із Surdophone						
			-3	-2	-1	0	+1	+2	+3
1	Цінова політика	16			+				
2	Точність перекладу	18				+			
3	Інтуїтивно зрозумілий інтерфейс	15				+			
4	Наявність додаткового функціоналу	18						+	
5	Точність розпізнавання природної мови	16				+			
6	Локалізація	20							+

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 4.12) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (табл. 4.11).

Таблиця 4.12

SWOT аналіз стартап-проекту

Сильні сторони (S): Висока точність перекладу Наявність додаткового функціоналу Наявність локалізації	Слабкі сторони (W): Цінова політика Відносна складність розробки
Можливості (O): Розширення функціоналу	Загрози (T): Недостатнє фінансування Поява конкурентів

На основі SWOT-аналізу розробимо альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи аналізуватимемо з точки зору строків та ймовірності отримання ресурсів (табл. 4.13).

Таблиця 4.13

Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Додавання функціоналу за певну плату	+	1-2 місяці
2	Реклама	Власні кошти, краудфандинг	1-2 місяці
3	Зменшення витрат на розробку (використовувати безкоштовні рішення)	+	3-4 тижні

Обрана альтернатива: Зменшення витрат на розробку завдяки використанню безкоштовних рішень.

4.4 Розроблення ринкової стратегії проекту

Для початку опишемо цільові групи потенційних споживачів.

Таблиця 4.14

Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Батьки	Висока зацікавленість	Попит середній – вище середнього	Середня	Середня складність
2	Викладачі	Середня зацікавленість	Попит середній	Відсутність на початку, подальше збільшення	Складність вищесереднього
Обидві групи вибрані як цільові так, як простота входу у сегмент з групою 1 простіша в реалізації і, в той же час, група 2 може надати додаткову рекламу продукту та, в перспективі, можливе введення додаткового функціоналу за окрему плату для групи 2.					

Для роботи в обраних сегментах ринку сформуємо базову стратегію розвитку та стратегію конкурентної поведінки.

Таблиця 4.15

Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентноспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Розширений функціонал	Реклама	Розширений функціонал	Стратегія диференціації

Таблиця 4.16

Визначення базової стратегії конкурентної поведінки

«Чи є прохід першопрохідцем на ринку»	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів	Чи буде компанія копіювати основні характеристики товару конкурента	Стратегія конкурентної поведінки
Так	Так	Ні	Монополізація

На основі вимог споживачів з обраних сегментів до постачальника (стартап компанії) та до продукту, а також в залежності від обраної базової стратегії розвитку та стратегії конкурентної поведінки розробимо стратегію позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.17

Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1	Точність, швидкість, надійність, безвідмовність	Диференціації	Точність, швидкість, надійність, безвідмовність	Точність, швидкість, надійність, безвідмовність

На основі даних, описаних вище можна зробити висновок, що розвиватися треба в напрямку ключових функцій, а саме: точності розпізнавання та перекладу і можливостях займатися вдома та розширення словнику.

4.5. Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.18

Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги над конкурентами, які пропонує товар
1	Швидкість		Комбінація ОС та мови розробки
2	Автоматизованість		
3	Використання БД		«Вшиті» засоби

Розробимо трирівневу маркетингову модель товару: уточнимо ідею продукту, його складові, особливості процесу його надання.

Таблиця 4.19

Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
Товар за задумом	Адаптивна система спілкування для дітей з вадами слуху		
Товар у загальному вигляді	Властивості/характеристики	М/Нм	Вр/Тх/Тл/Е/Ор
	Зручність використання	-	Висока
	Наявність української мови жестів	-	Так
	Кількість мов підтримки	Шт.	1
	Вартість	Грн	0
	Наявність розширюваності	-	Наявна
	Програма пройшла тестування та повністю відповідає визначеним головним вимогам ринку.		
	Доступна за адресою в мережі Інтернет		
	Марка: Pingwin, назва: Перекладач		

Продовження таблиці 4.19

Товар з підкріпленням	До продажу: наявна повна документація, акції на придбання декількох ліцензій
	Після продажу: підтримка з боку розробника та адміністраторів сайту
Проект буде захищено від копіювання реєстрацією назви програми, створення заявки на отримання патенту на винахід.	

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар, яке передбачає аналіз ціни на товари- аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.20). Аналіз проводиться експертним методом.

Таблиця 4.20

Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Замінники відсутні	Більшість аналогів безкоштовні, з платним додатковим функціоналом в районі 200-500 грн.	Нижче середнього Середні Вище середнього	На даний момент додаток повністю безкоштовний для користувачів

Наступним кроком визначимо оптимальну систем збуту, в межах якого приймається рішення.

Таблиця 4.21

Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Придбання реклами на сайті	Електронний вигляд. Доступ для купівлі через специфічні платформи Google Adwords та аналоги	Виробник споживач	Офіційний сайт виробника із сторінкою прайсингу та придбання

Розробимо концепцію маркетингових комунікацій.

Таблиця 4.22

Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	-	Лінія зворотного зв’язку на сайті, онлайн- чат, електронна пошта (неформальні)	Якість послуги Доступність та коректність інформації, шо надається	Поширення інформації про рішення, його функціонал та переваги перед конкурентами	Звернення засноване на виділені якісних характеристик розробки

Висновки до розділу

Під час аналізу було виявлено, що ринок на даний момент зростає, наявний попит на системи такого плану та відносно невеликий рівень конкуренції та майже відсутні обмеження для входу на ринок. Враховуючи ці фактори можна зробити висновок, що на даний момент існує можливість ринкової комерціалізації проекту.

На даний момент існує декілька схожих рішень, але жодне із них не є монополістом на ринку. Так, як проект має переваги над уже існуючими системами, ринок для входження стартап-продукту є привабливим.

Альтернативним варіантом впровадження є розробка специфічного функціоналу за додаткові кошти.

Проаналізувавши фактори, можна дійти до висновку, що подальша реалізація проекту є доцільною.

ВИСНОВКИ

У ході виконання магістерської дисертації було розглянуто проблему розрахунку академічної заборгованості з надання освітніх послуг в Україні. Проведений аналіз аналогічних технологій та експериментальним шляхом підібраний найоптимальніший набір рішень для створення програмного забезпечення.

У розділі аналізу рівня автоматизації обліку академічної заборгованості з надання освітніх послуг було розглянуто академічну заборгованість з надання освітніх послуг у вищих навчальних закладах, електронний документообіг, переваги та недоліки існуючих аналогів та сформульовано необхідність створення нової системи.

У розділі вибору технологій розробки розглянуто сучасні операційні системи, мови програмування та програмні рішення для розробки власного програмного забезпечення. Визначено вимоги до системи.

У розділі практичного застосування наведено приклади роботи сервісу та описується процес взаємодії користувача та розробника з додатком.

Розроблене програмне забезпечення повинно у повній мірі задовольнити усі потреби, що існують на сьогоднішній день у сфері розрахунку академічної заборгованості з надання освітніх послуг для вищих навчальних закладів України.

Проведено аналіз ринку та можливостей виходу стартап-проекту даної магістерської дисертації на цей ринок. Розроблений програмний продукт у вигляді гнучкого роботизованого сервісу повинен зайняти лідерські позиції та створити монополію на ринку. Виявлені перспективи виходу на ринок та доцільність подальшого розвитку проекту. Розроблено ринкову стратегію та маркетингову програму стартап-проекту. У розділі проаналізовано поточну ситуацію на ринку, розроблено стратегії та маркетинговий плани для впровадження даного рішення.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Закон України про освіту https://zakon.rada.gov.ua/laws/show/2145-19/ed20170905#n27

2. Коломоєць С.О. Використання роботизованих сервісів для автоматизації управління ВНЗ. International Electronic Scientific and Practical Journal «WayScience» №2 (4). Дніпро, 2019.

3. Коломоец С.А. Автоматизация расчета академической задолженности в высших учебных заведениях. XV Международная научно-практическая конференция «Перспективные вопросы мировой науки. София, Болгария, 2019 с. 51-55.

4. https://mon.gov.ua/storage/app/media/regulatorna_dijalnist/2018/05/2/2polozhennya-pro-perevedennya-17012018-roku.pdf

5. https://assets.omron.eu/downloads/brochure/engb/v16/robotics_services_overview_brochure_engb.pdf

6. https://www.springer.com/journal/10696

7. https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D1%96%D1%97_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D0%B1%D1%96%D0%B3%D1%83

ДОДАТКИ

ДОДАТОК А СТАТТЯ: «ВИКОРИСТАННЯ РОБОТИЗОВАНИХ СЕРВІСІВ ДЛЯ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ВНЗ»

ВИКОРИСТАННЯ РОБОТИЗОВАНИХ СЕРВІСІВ ДЛЯ АВТОМАТИЗАЦІЇ УПРАВЛІННЯ ВНЗ

Коломоєць С.О., Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», студент 6 курсу Факультету інформатики та обчислювальної техніки (ФІОТ), кафедри Технічної кібернетики (ТК), спеціальності «Інформаційні системи та технології»

ИСПОЛЬЗОВАНИЕ РОБОТИЗИРОВАННЫХ СЕРВИСОВ ДЛЯ АВТОМАТИЗАЦИИ УПРАВЛЕНИЯ ВУЗ

Коломоец С.А., Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», студент 6 курса Факультета информатики и вычислительной техники (ФИВТ), кафедры Технической кибернетики (ТК), специальности «Информационные системы и технологии»

USING OF ROBOTIZED SERVICES FOR AUTOMATION OF MANAGEMENT OF UNIVERSITY

Kolomoiets Serhii, National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», 6th year student of Faculty of Informatics and Computer Science (FICS), Department of Technical Cybernetics (TC), specialty «Information systems and technologies»

Постановка проблеми. На сьогоднішній день відбувається перехід від звичної форми праці до її автоматизації, що в свою чергу економить час, ресурси, кошти та робить оптимізацію використання людських ресурсів. Але через те, що галузь освіти направлена на надання освітніх послуг, бюрократичний апарат не має достатнього державного фінансування. Звичайно що технічні ВНЗ мають доступ до безкоштовних підписок на всі продукти для студентів на момент їх навчання від компаній як Microsoft, або суттєві знижки від компаній з перевірки робіт на плагіат як Unicheck, а також

спеціальні ціні на сервіси за підпискою від компаній як Apple, але коли справа стосується бюрократичної частини, то уся робота, зазвичай, виконується руками, працівників ВНЗ. У сьогоднішніх реаліях новий уряд оголосив курс на автоматизацію управління, спрощення бюрократії та знайдення нових шляхів в управлінні різноманітними структурами, в якій сфера освіти потребує не меншої уваги ніж усі інші.

Аналіз останніх досліджень і публікацій. Згідно відкритої інформації, на сьогоднішній день в Україні не існує роботизованих сервісів, які б дозволили автоматизувати управління ВНЗ. Якщо подібні рішення і існують, то лише у межах певного ВНЗ, що робить їх недоступними для загального аналізу та використання. Зазвичай подібними продуктами не прийнято ділитися, тому що вони відповідають вимогам лише окремо взятого навчального закладу та зроблені на основі знань студентів що там навчаються і не мають комерційної складової та не пристосовані для загального використання, або навчальний заклад зовсім не має засобів для автоматизації документообігу.

Мета і завдання статті. Метою даної статті є визначення сфер у житті ВНЗ в яких необхідно підвищити ефективність роботи за допомогою використання автоматизованого управління.

Згідно поставленої мети можна виділити наступні завдання: сформулювати вимоги до системи управління, проаналізувати сфери застосування подібних систем, на прикладі розробленого модулю системи автоматизації управління довести доцільність використання додатків такого типу.

Виклад основного матеріалу. Основним критерієм для вибору технологій створення сервісів є підтримка розробленого продукту усіма пристроями, що використовують у подібній сфері. Зважаючи на технології, що

використовують у ВНЗ потрібно обрати оптимальні технології розробки сервісів автоматизації управління.

Таблиця 1. Обрані технології для розробки роботизованого сервісу для автоматизації управління

№	Вимоги	Обрані рішення	Аргументація вибору
1	Операційна система	Windows	Саме ця операційна система використовується усіма ВНЗ
2	Мова програмування	C# (.NET)	Мова C# є об'єктно-орієнтованою та розроблена під ОС Windows
3	Підтримка версій	.NET 4 - 4.7.2.	Технологія .NET 4 підтримується починаючи з Windows 7
4	Використання баз даних	API + локальна БД	Можливість використовувати власні API або локальну БД якщо не має доступу до мережі Internet

Проаналізувавши задачі поставлені перед апаратом управління ВНЗ можна виділити наступні сфери застосування:

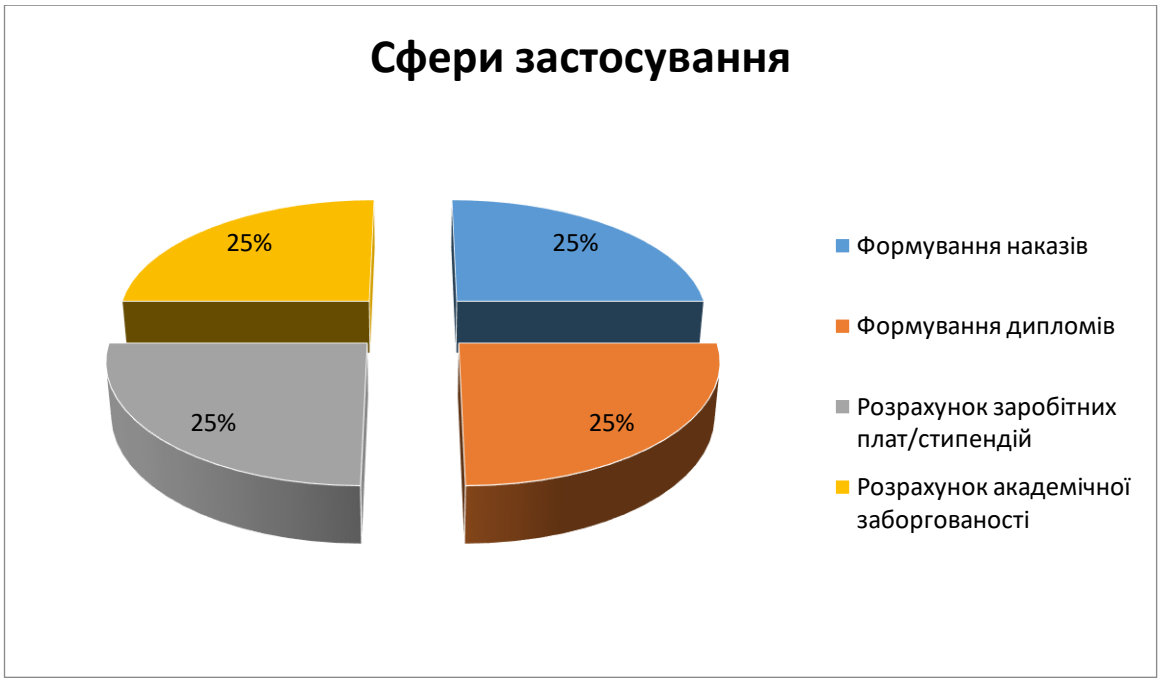


Рисунок 1. Сфери застосування систем автоматизації управління ВНЗ

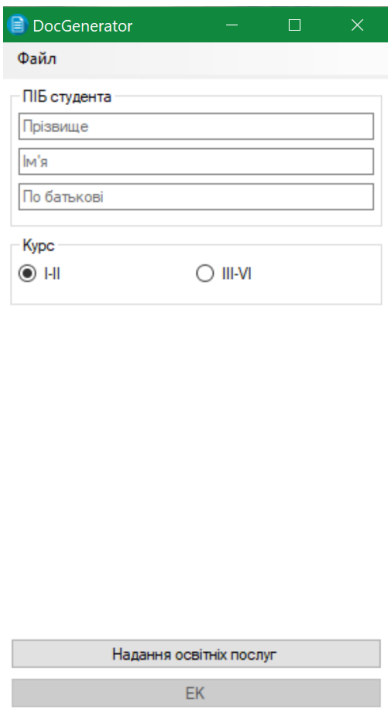


Рисунок 2. Початкове вікно розробленого додатку для розрахунку академічної заборгованості з надання освітніх послуг

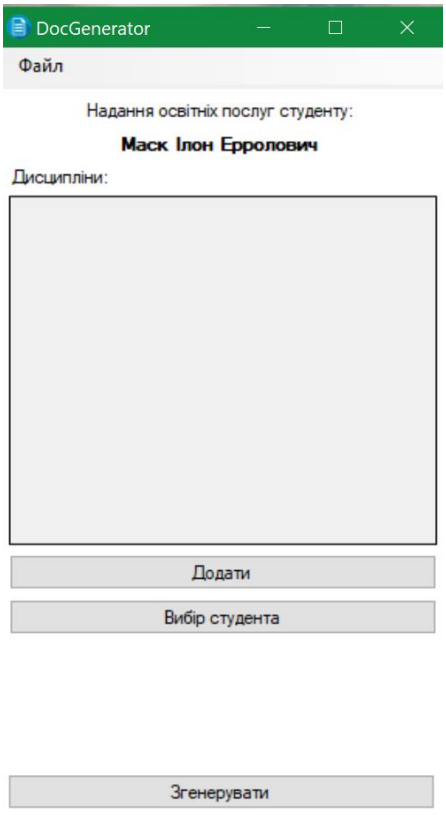


Рисунок 3. Персональне вікно для розрахунку академічної заборгованості окремого студента

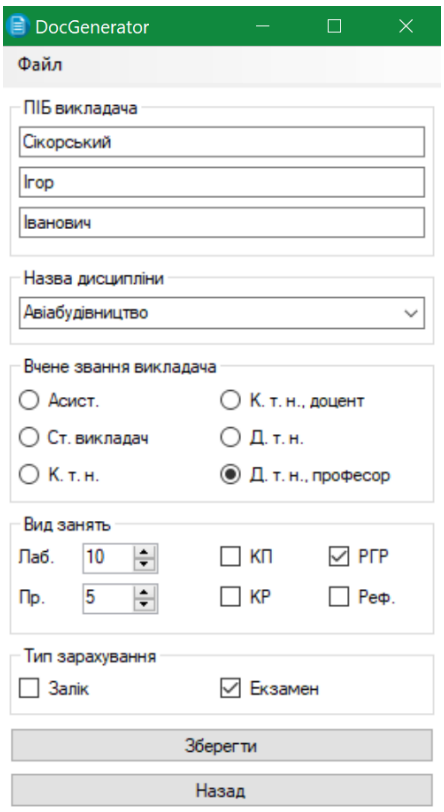


Рисунок 4. Вікно «Викладач»

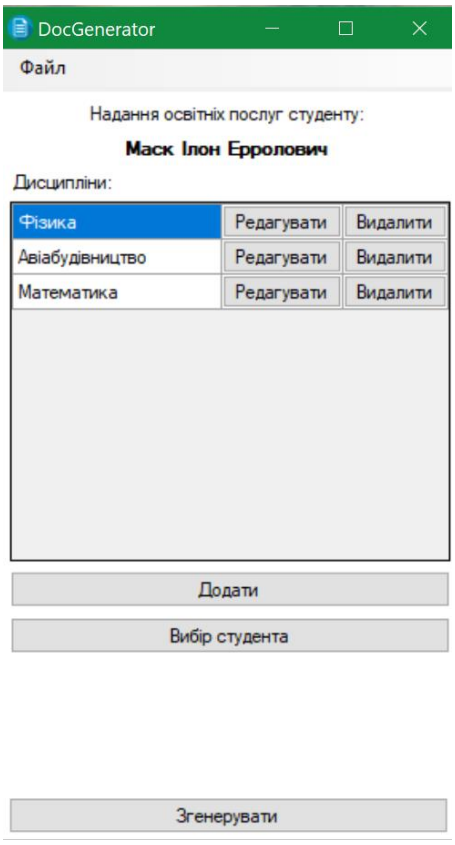


Рисунок 5. Заповнене вікно «Надання освітніх послуг студенту»

Висновки та перспективи подальшого розвитку. У середньому подібна робота з формування документів для студентів, які мають академічну заборгованість складає від кількох днів до тижня, в залежності від кількості студентів. Розроблений додаток може бути використаним будь-яким ВНЗ для формування документації для даного типу роботи. Технології підібрані універсально, що робить рішення кросплатформним, тобто проблеми використання не виникнуть ні на яких апаратах. Незважаючи на те, що дане рішення є лише модулем відносно повноцінної системи автоматизації управління ВНЗ, воно продемонструвало себе саме так, як і очікувалось, використання часу, ресурсів та завантаженість персоналу знизилося з тижня до години. Якщо в подальшому реалізувати інші модулі для автоматизації управління, то це дозволить розвантажити апарат управління та покращить якість надання освітніх послуг.

ДОДАТОК Б СТАТТЯ: «АВТОМАТИЗАЦИЯ РАСЧЕТА АКАДЕМИЧЕСКОЙ ЗАДОЛЖЕННОСТИ В ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЯХ»

Коломоец Сергей Алексеевич

Национальный технический университет Украины «Киевский политехнический институт имени Игоря Сикорского», Украина

Автоматизация расчета академической задолженности в высших учебных заведениях

Академическая задолженность — это неудовлетворительные результаты промежуточной аттестации по одному или нескольким учебным предметам, курсам, дисциплинам (модулям) или не прохождение промежуточной аттестации при отсутствии уважительных причин [1]. Для продолжения обучения учащийся обязан ликвидировать академическую задолженность пересдав определённый предмет. В случае отчисления студента перед защитой бакалаврской работы или магистерской диссертации пересдача защиты диплома перед экзаменационной комиссией считается академической задолженностью.

На сегодняшний день, согласно открытым источникам информации, не существует готовых решений проблемы автоматизации расчета академической задолженности по предоставлению образовательных услуг. Проблематика данного вопроса заключается в использовании труда сотрудников ВУЗ и неэффективном использовании ресурсов ВУЗ.

Система автоматизации документооборота — это автоматизированная многопользовательская система, сопровождающая процесс управления работой иерархической организации с целью обеспечения выполнения этой организацией своих функций [2].

Применив систему автоматизации документооборота к данной проблеме возможно сократить время и количество ресурсов. На практическом опыте

было проанализировано расчет академической задолженности по факультету (870 студентов всех форм обучения). При использовании традиционных средств расчета академической задолженности занимаемое время составило ≈ неделю. С помощью внедрения сервиса по расчёту академической задолженности удалось оптимизировать ту же работу в течении 2 часов.

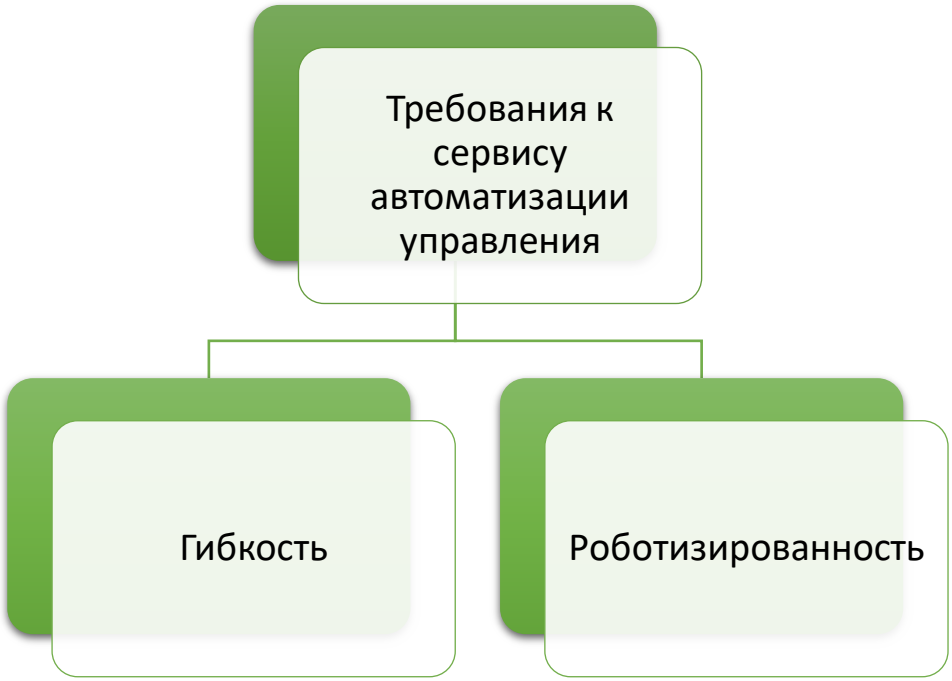


Рисунок 1. Требования к сервису автоматизации управления

Основным критерием для выбора технологий создания сервиса является поддержка разработанного продукта всеми устройствами, использующими в сфере предоставления образовательных услуг. Опираясь на технологии, использующиеся в ВУЗ были выбраны оптимальные технологии разработки для сервиса автоматизации управления академической задолженностью.

Таблица 1. Выбранные параметры для разработки сервиса

Операционная система	Версия ОС	Язык программирования	Версия языка разработки	База данных
Windows	7 и новее	C#	.NET 4.0 и новее	MS Access

Предложенное решение проблемы автоматизации расчета академической задолженности в высших учебных заведениях в виде разработанного гибкого роботизированного сервиса:

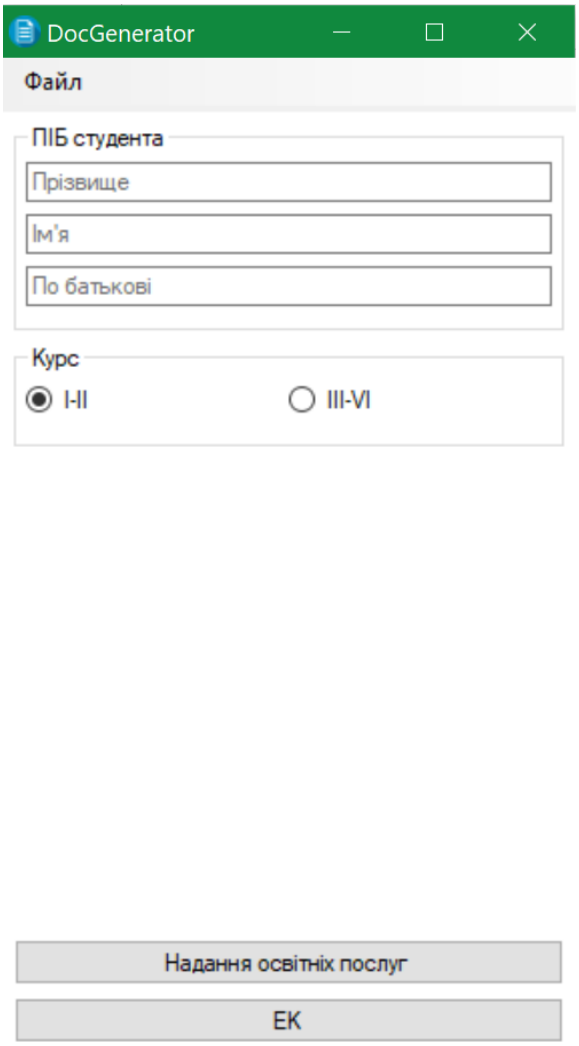


Рисунок 2. Начальное окно сервиса для выбора студента которому будут предоставляться образовательные услуги

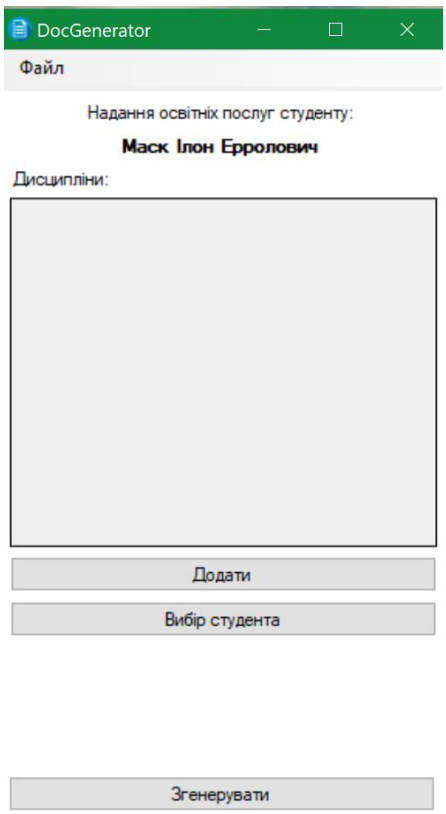


Рисунок 3. Меню дисциплин выбранного студента

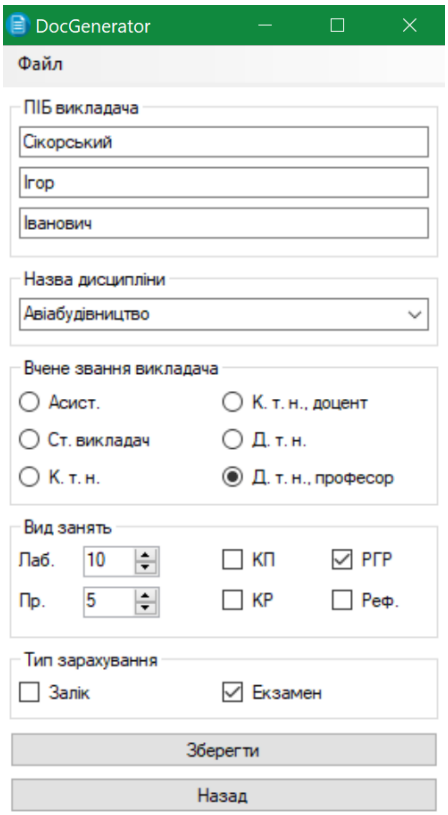


Рисунок 4. Меню преподавателя предоставляющего образовательную услугу

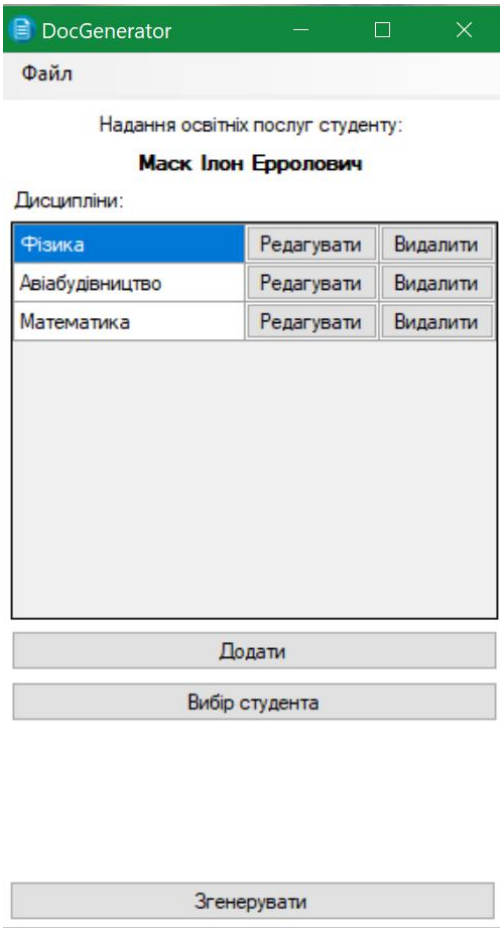


Рисунок 5. Заполненное меню дисциплин для выбранного студента

Литература

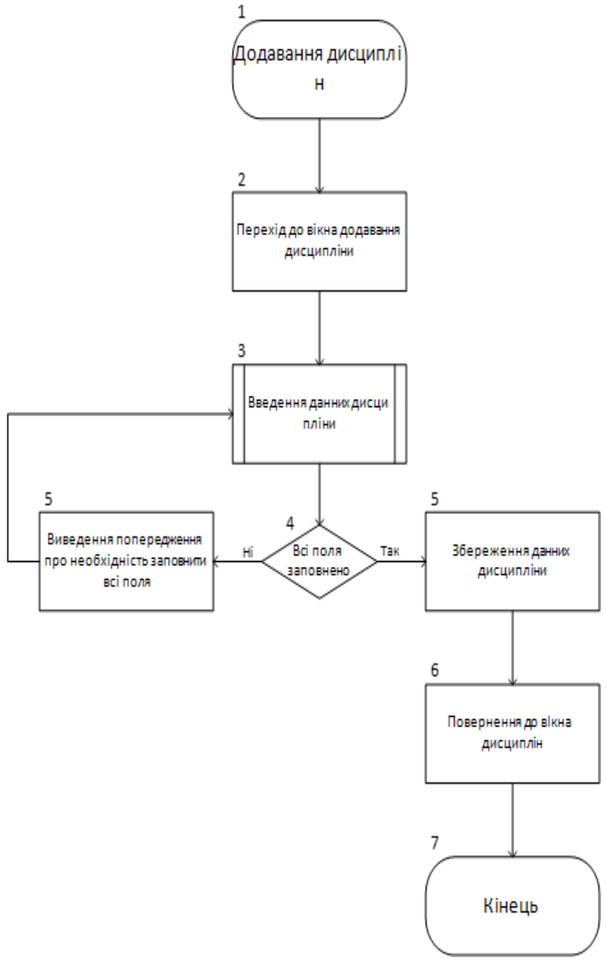
1. https://tti.wikia.org/ru/wiki/%D0%90%D0%BA%D0%B0%D0%B4%D0%B5%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%BE%D0%BB%D0%B6%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C
2. https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%BE%D0%B1%D0%BE%D1%80%D0%BE%D1%82%D0%B0

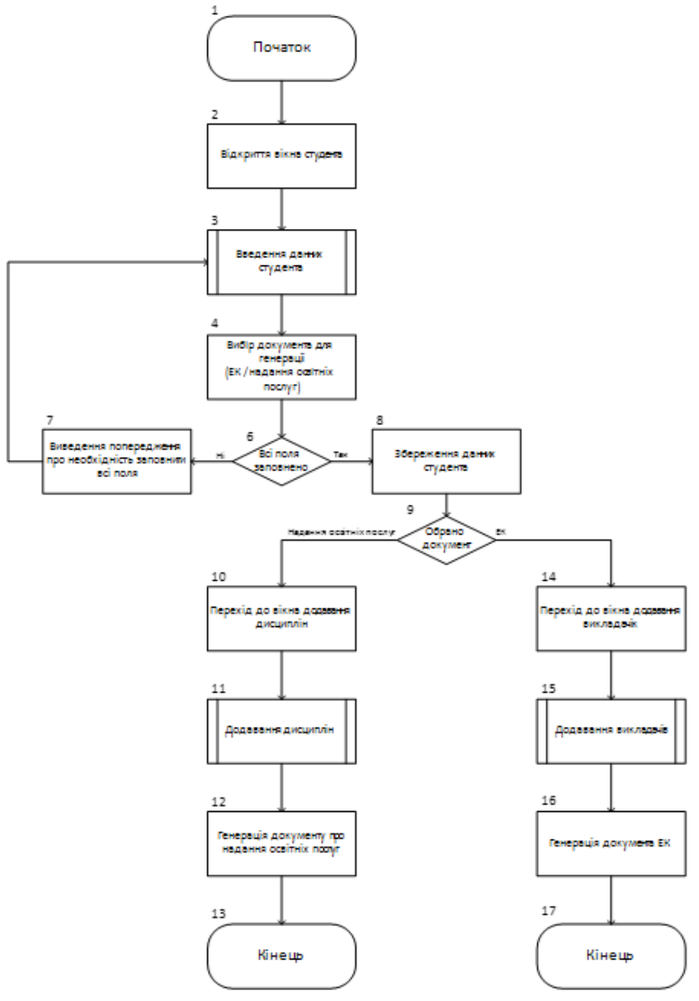
ДОДАТОК В: ПЕРЕВІРКА НА СПІВПАДІННЯ

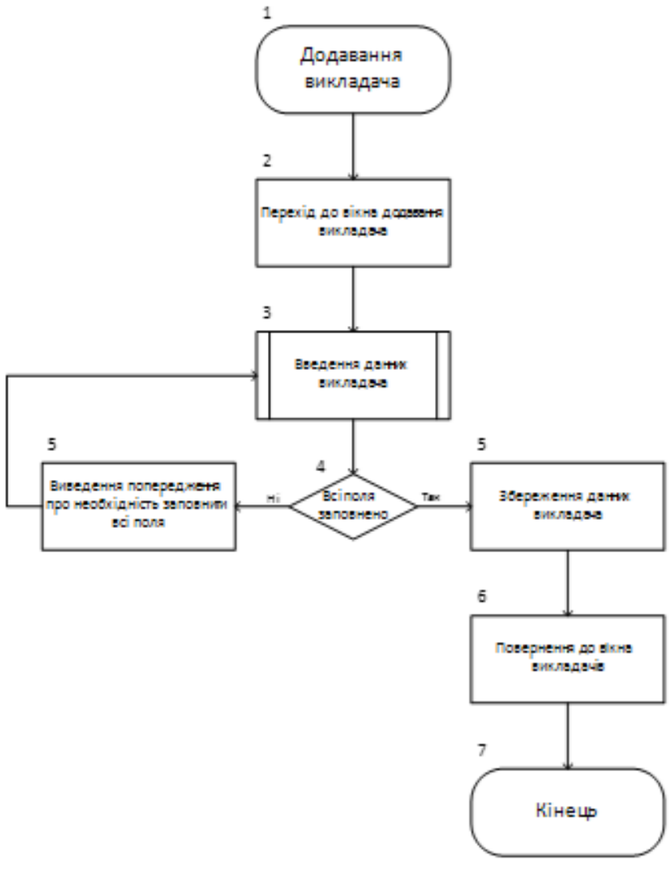
ДОДАТОК Г: ПЕРЕЛІК ІЛЮСТРАЦІЙНОГО МАТЕРІАЛУ

ПЛАКАТ 1: ЛОГІЧНА СХЕМА РОБОТИ СЕРВІСУ

ЛОГІЧНА СХЕМА РОБОТИ СЕРВІСУ





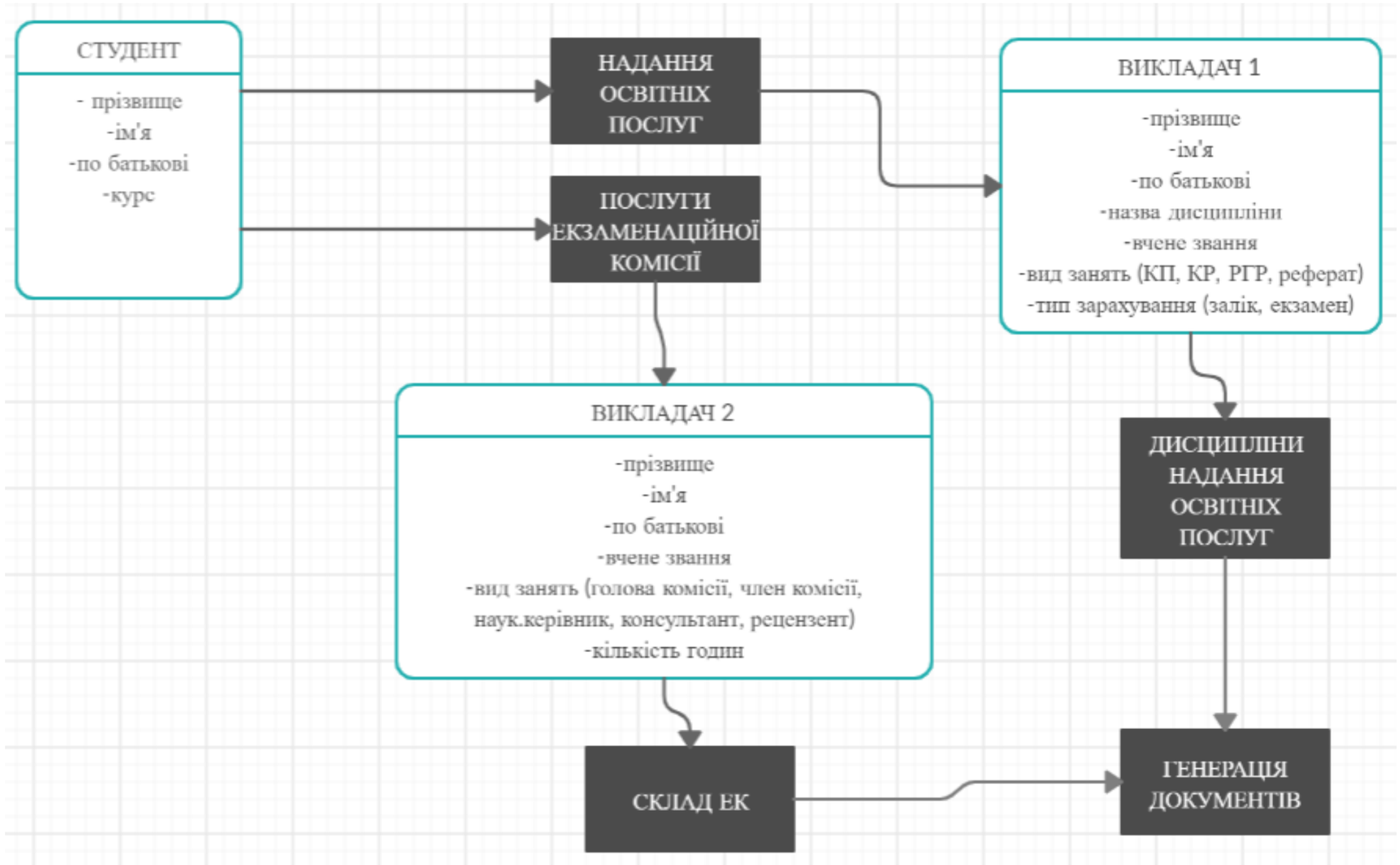


Демонстраційний плакат №1
до магістерської дисертації на тему
«Гнучкий роботизований сервіс розрахунку академічної заборгованості з надання освітніх послуг»

Розробив: Коломоєць С.О.
Прийняв: к.т.н., доцент Остапченко К.Б.

ПЛАКАТ 2: СХЕМА РОБОТИ БАЗИ ДАНИХ

СХЕМА РОБОТИ БАЗИ ДАНИХ

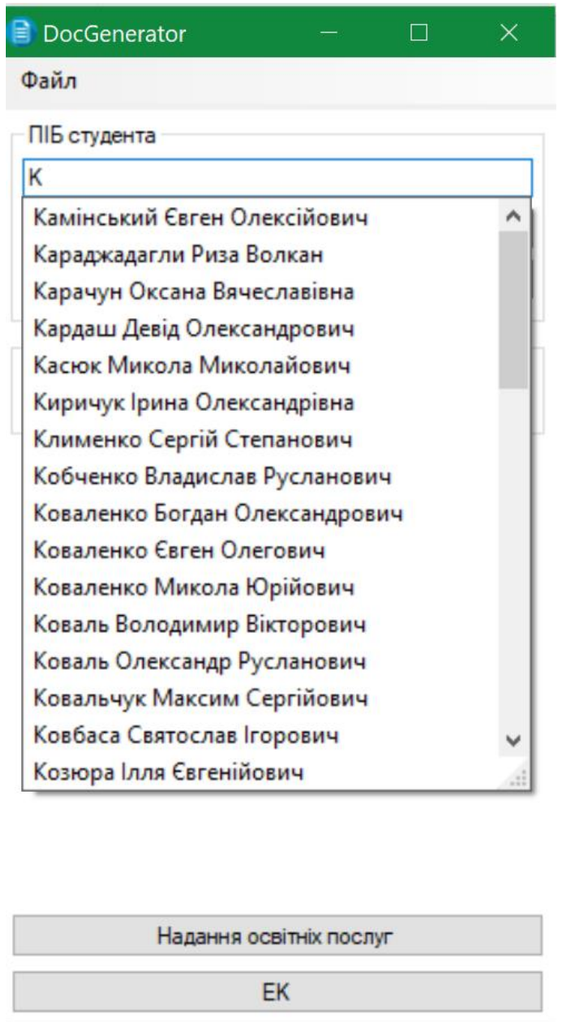


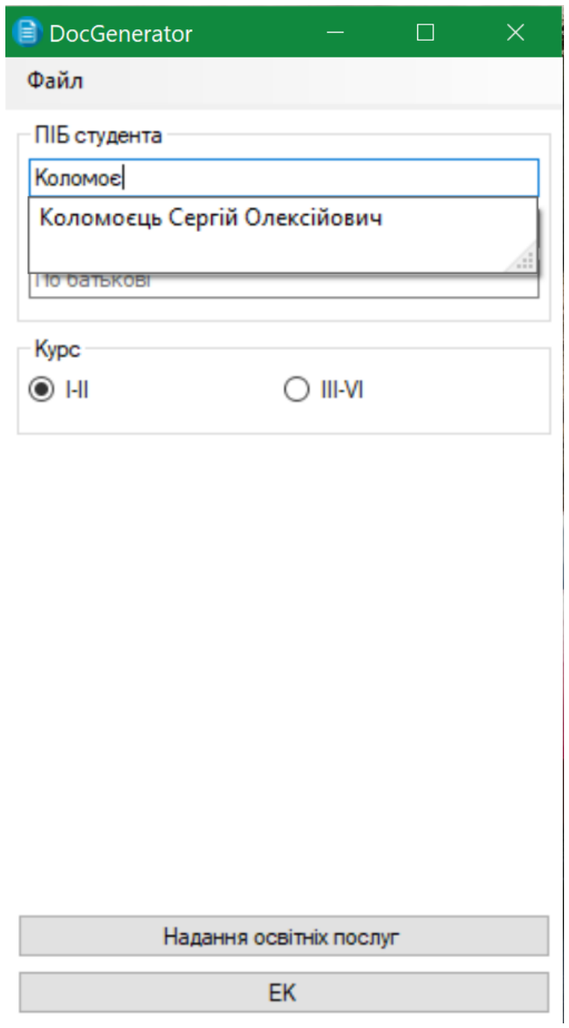
Демонстраційний плакат №2
до магістерської дисертації на тему
«Гнучкий роботизований сервіс розрахунку академічної заборгованості з надання освітніх послуг»

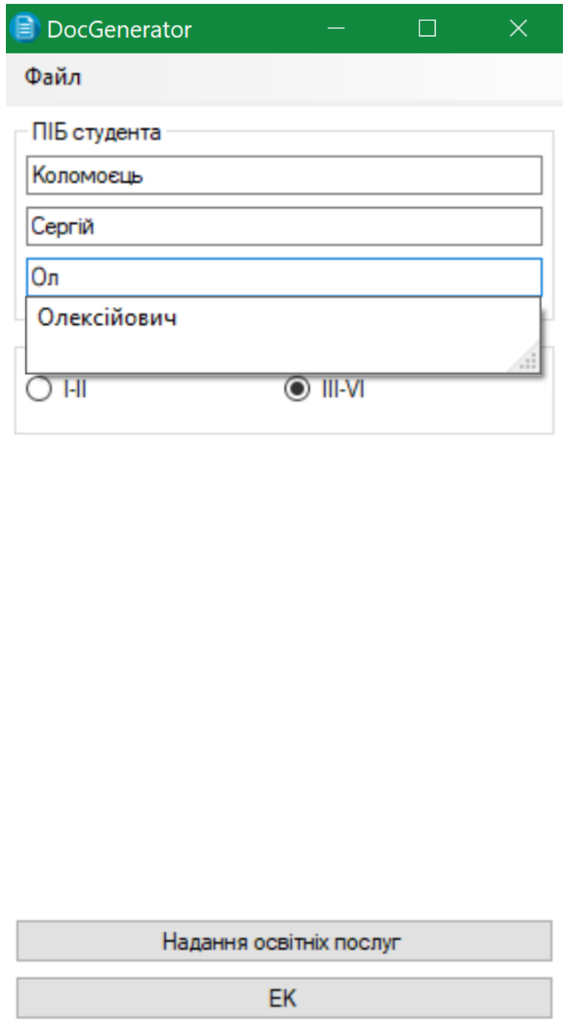
Розробив: Коломоєць С.О.
Прийняв: к.т.н., доцент Остапченко К.Б.

ПЛАКАТ 3: СКРІНШОТИ РОБОТИ СЕРВІСУ

СКРІНШОТИ РОБОТИ СЕРВІСУ





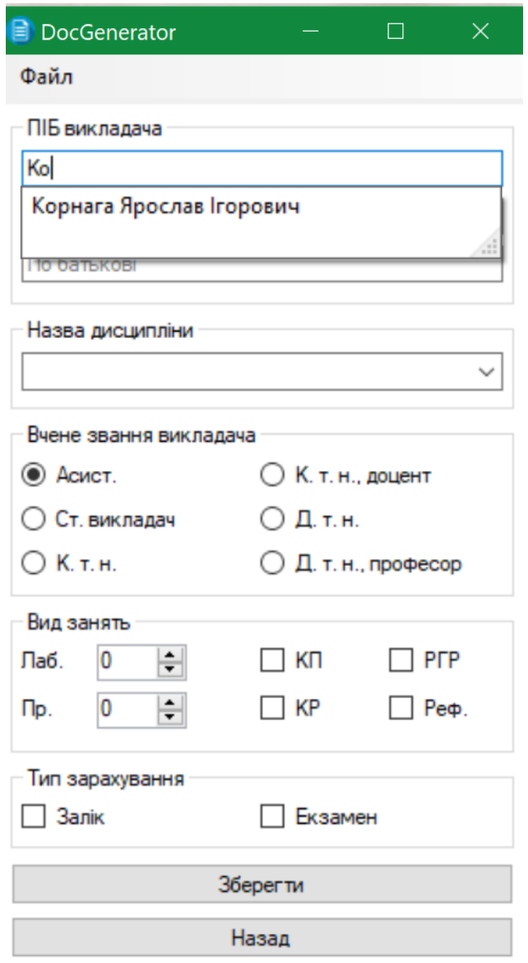


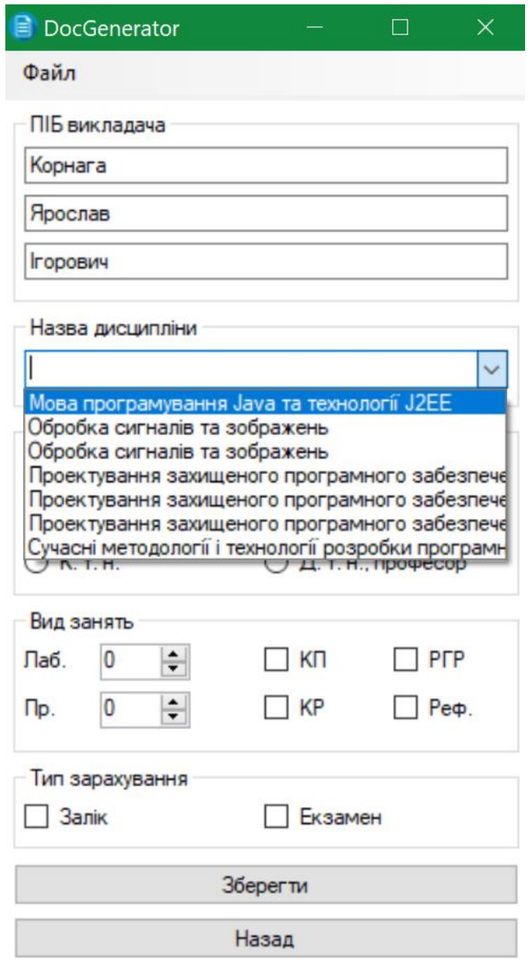
Демонстраційний плакат №3
до магістерської дисертації на тему
«Гнучкий роботизований сервіс розрахунку академічної заборгованості з надання освітніх послуг»

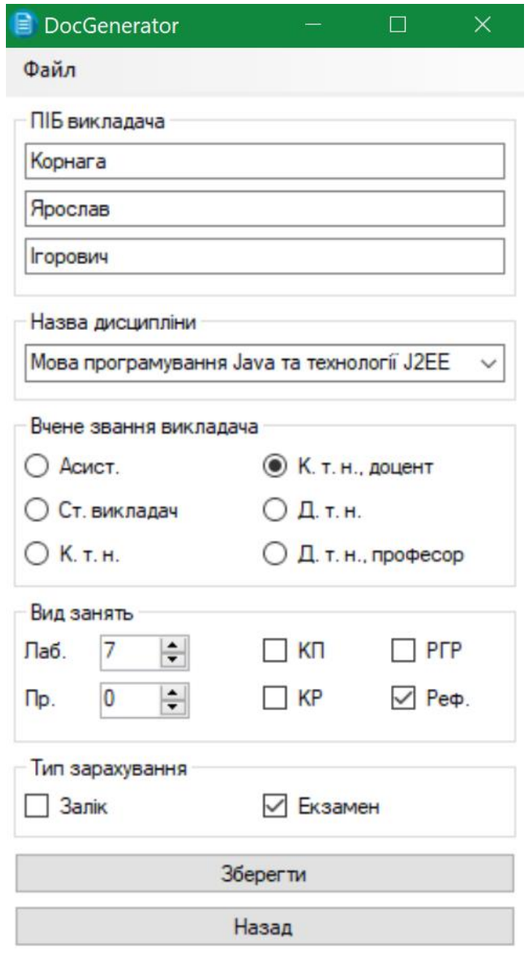
Розробив: Коломоєць С.О.
Прийняв: к.т.н., доцент Остапченко К.Б.

ПЛАКАТ 4: СКРІНШОТИ РОБОТИ СЕРВІСУ

СКРІНШОТИ РОБОТИ СЕРВІСУ







Демонстраційний плакат №4
до магістерської дисертації на тему
«Гнучкий роботизований сервіс розрахунку академічної заборгованості з надання освітніх послуг»

Розробив: Коломоєць С.О.
Прийняв: к.т.н., доцент Остапченко К.Б.

ДОДАТОК Д: ЛІСТИНГ

Лістинг коду:

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DocsGenerator
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

Process.cs

```
using System.Collections.Generic;
using System.IO;
using Spire.Xls;
using System.Drawing;

namespace DocsGenerator.Classes
{
    class Process
    {
        // Singleton interface
        private static Process instance;
        private Process() {}
        public static Process GetInstance()
        {
            if (instance == null)
            {
                instance = new Process();
            }
            return instance;
        }

        // Program proccess
        private Student student;
        private readonly List<IDocument> documents = new List<IDocument>
        {
            new Act(),
            new Calculation(),
            new Contract()
        };
        private readonly EK EKdocument = new EK();
        private readonly List<Discipline> disciplines = new List<Discipline>();
        private readonly List<Lecturer> comissionMembers = new List<Lecturer>();
        private readonly string excelPath =
$"{Directory.GetCurrentDirectory()}\\history.xlsx";
```

```
        public void Generate(string path)
        {
            foreach (IDocument doc in documents)
            {
                doc.Generate(path, this.GetStudentName());
                System.Threading.Thread.Sleep(500);
            }
            System.Diagnostics.Process.Start(path +
$"\\{this.GetStudentName()}.docx");
            AddExcel();
        }

        public void GenerateEK(string path)
        {
            EKdocument.Generate(path, this.GetStudentName());
            System.Threading.Thread.Sleep(500);
            System.Diagnostics.Process.Start(path +
$"\\{this.GetStudentName()}.docx");
        }

        public void AddExcel()
        {
            Workbook workbook = new Workbook();
            if (File.Exists(excelPath))
            {
                workbook.LoadFromFile(excelPath);
            }
            else
            {
                workbook.Worksheets.Clear();
                workbook.Worksheets.Add("Students");
                workbook.Worksheets[0].Range["A1:D1"].Style.Color =
Color.GreenYellow;
                workbook.Worksheets[0].Range["A1"].Value = "ПІБ студента";
                workbook.Worksheets[0].Range["B1"].Value = "Група";
```

```
            workbook.Worksheets[0].Range["C1"].Value = "Дисципліна";
            workbook.Worksheets[0].Range["D1"].Value = "Викладач";
        }
        Worksheet sheet = workbook.Worksheets[0];

        foreach (Discipline discipline in disciplines)
        {
            sheet.InsertRow(2);
            sheet.Range["A2"].Value = student.GetFullName();
            sheet.AutoFitColumn(1);

            sheet.Range["B2"].Value = student.group;
            sheet.AutoFitColumn(2);

            sheet.Range["C2"].Value = discipline.name;
            sheet.AutoFitColumn(3);

            sheet.Range["D2"].Value = discipline.lect.GetFullName();
            sheet.AutoFitColumn(4);
        }
        workbook.SaveToFile(excelPath);
        workbook.Dispose();
    }

    public void SetStudent(Student student)
    {
        this.student = student;
    }
    public string GetStudentName()
    {
        return student.GetFullName();
    }
    public bool GetStudentIsElderCourse()
    {
        return student.isElderCourse;
```

```
}
public void AddDiscipline(Discipline discipline)
{
    disciplines.Add(discipline);
}
public void DeleteDiscipline(int Index)
{
    disciplines.RemoveAt(Index);
}
public void EditDiscipline(int Index, Discipline discipline)
{
    disciplines[Index] = discipline;
}
public List<Discipline> GetDisciplines()
{
    return disciplines;
}
public void AddComissionMember(Lecturer lecturer)
{
    comissionMembers.Add(lecturer);
}
public void DeleteComissionMember(int Index)
{
    comissionMembers.RemoveAt(Index);
}
public void EditComissionMember(int Index, Lecturer lecturer)
{
    comissionMembers[Index] = lecturer;
}
public List<Lecturer> GetComissionMembers()
{
    return comissionMembers;
}
public double GetTotalCost()
{
```

```
    double sumcost = 0;
    foreach (Discipline discipline in disciplines)
    {
        sumcost += discipline.GetCost();
    }
    return sumcost;
}
public double GetTotalHours()
{
    double sumhours = 0;
    foreach (Discipline discipline in disciplines)
    {
        sumhours += discipline.GetHours();
    }
    return sumhours;
}

public double GetComissionTotalCost()
{
    double sumcost = 0;
    foreach (Lecturer comissionMember in comissionMembers)
    {
        sumcost += comissionMember.GetMemberCost();
    }
    return sumcost;
}

public double GetComissionTotalHours()
{
    double sumhours = 0;
    foreach (Lecturer comissionMember in comissionMembers)
    {
        sumhours += comissionMember.GetMemberHours();
    }
    return sumhours;
```

```
        }

        public double GetMemberTypeCost(ComType comType)
        {
            double cost = 0;
            foreach (Lecturer comissionMember in comissionMembers)
            {
                if (comissionMember.MemberType == comType)
                {
                    cost += comissionMember.GetMemberCost();
                }
            }
            return cost;
        }

        public double GetMemberTypeHours(ComType comType)
        {
            double hours = 0;
            foreach (Lecturer comissionMember in comissionMembers)
            {
                if (comissionMember.MemberType == comType)
                {
                    hours += comissionMember.GetMemberHours();
                }
            }
            return hours;
        }

        public void OpenExcel()
        {
            System.Diagnostics.Process.Start(excelPath);
        }
    }
}
```

API.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Net;
using Newtonsoft.Json;


namespace DocsGenerator.Classes
{
    class API
    {
        protected static string faculId = "20";
        protected static string baseUrl = "http://apis.kpi.ua/api/";
        protected static string apiKey = "4LmX7GGUrp9L8FZ2";
        public static List<Student> GetStudents(string Name, string Surname, string
Fathername)
        {
            if (Name == "Ім'я")
            {
                Name = "";
            }
            if (Surname == "Прізвище")
            {
                Surname = "";
            }
            if (Fathername == "По батькові")
            {
                Fathername = "";
            }


            string url = baseUrl +
$"search/student?facultyId={faculId}&firstName={Uri.EscapeDataString(Name)}&lastName=
{Uri.EscapeDataString(Surname)}&middleName={Uri.EscapeDataString(Fathername)}";
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
            request.Headers.Add(HttpRequestHeader.Authorization, apiKey);
            request.AutomaticDecompression = DecompressionMethods.GZip |
DecompressionMethods.Deflate;
```

```
            List<Student> students = new List<Student>();


            try
            {
                using (HttpWebResponse response =
(HttpWebResponse)request.GetResponse())
                {
                    if (response.StatusCode == HttpStatusCode.OK)
                    {
                        using (Stream stream = response.GetResponseStream())
                        using (StreamReader reader = new StreamReader(stream))
                        {
                            var studentsAsJson = reader.ReadToEnd();
                            dynamic obj =
JsonConvert.DeserializeObject<List<StudentJson>>(studentsAsJson);
                            foreach (StudentJson student in obj)
                            {
                                students.Add(student.GetStudent());
                            }
                        }
                    }
                }
                return students;
            }
            catch
            {
                students.Clear();
                return students;
            }
        }


        public static List<Lecturer> GetLecturers(string Name, string Surname, string
Fathername)
        {
            if (Name == "Ім'я")
            {
```

```
                Name = "";
            }
            if (Surname == "Прізвище")
            {
                Surname = "";
            }
            if (Fathername == "По батькові")
            {
                Fathername = "";
            }
            string url = baseUrl +
$"search/employee?facultyId={faculId}&firstName={Uri.EscapeDataString(Name)}&lastName
={Uri.EscapeDataString(Surname)}&middleName={Uri.EscapeDataString(Fathername)}";
            HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
            request.Headers.Add(HttpRequestHeader.Authorization, apiKey);
            request.AutomaticDecompression = DecompressionMethods.GZip |
DecompressionMethods.Deflate;
            List<Lecturer> lecturers = new List<Lecturer>();


            try
            {
                using (HttpWebResponse response =
(HttpWebResponse)request.GetResponse())
                {
                    if (response.StatusCode == HttpStatusCode.OK)
                    {
                        using (Stream stream = response.GetResponseStream())
                        using (StreamReader reader = new StreamReader(stream))
                        {
                            var studentsAsJson = reader.ReadToEnd();
                            dynamic obj =
JsonConvert.DeserializeObject<List<LecturerJson>>(studentsAsJson);
                            foreach (LecturerJson lecturer in obj)
                            {
                                lecturers.Add(lecturer.GetLecturer());
                            }
                        }
```

```
                }
            }
            return lecturers;
        }
        catch
        {
            lecturers.Clear();
            return lecturers;
        }
    }

    public static List<string> GetDisciplines(string employeeId) {
        string url = baseUrl + $"teacher-
disciplines?employeeId={Uri.EscapeDataString(employeeId)}";
        HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
        request.Headers.Add(HttpRequestHeader.Authorization, apiKey);
        request.AutomaticDecompression = DecompressionMethods.GZip |
DecompressionMethods.Deflate;
        List<string> DisciplineNames = new List<string>();

        try
        {
            using (HttpWebResponse response =
(HttpWebResponse)request.GetResponse())
            {
                if (response.StatusCode == HttpStatusCode.OK)
                {
                    using (Stream stream = response.GetResponseStream())
                    using (StreamReader reader = new StreamReader(stream))
                    {
                        var disciplinesAsJson = reader.ReadToEnd();
                        dynamic obj =
JsonConvert.DeserializeObject<List<DisciplineJson>>(disciplinesAsJson);
                        foreach (DisciplineJson discipline in obj)
                        {
                            DisciplineNames.Add(discipline.GetDisciplineName());
                        }
```

```
                    }
                }
            }
            return DisciplineNames;
        }
        catch
        {
            DisciplineNames.Clear();
            return DisciplineNames;
        }
    }
}
```

Person.cs

```
namespace DocsGenerator.Classes
{
    class Person
    {
        public string Name, Surname, FatherName;

        public Person(string Name, string Surname, string FatherName)
        {
            this.Name = Name;
            this.Surname = Surname;
            this.FatherName = FatherName;
        }
        public string GetFullName()
        {
            return $"{Surname} {Name} {FatherName}";
        }
    }
}
```

Student.cs

```
namespace DocsGenerator.Classes
{
    class Student : Person
    {
        public bool isElderCourse;
        public string group;

        public Student(string Name, string Surname, string FatherName, bool
isElderCourse, string group) : base(Name, Surname, FatherName)
        {
            this.isElderCourse = isElderCourse;
            this.group = group;
        }
    }
    class StudentJson
    {
        public string studentId, facultyName, groupName, firstName, middleName,
lastName;
        public StudentJson(string studentId, string facultyName, string groupName,
string firstName, string middleName, string lastName)
        {
            this.studentId = studentId;
            this.facultyName = facultyName;
            this.groupName = groupName;
            this.firstName = firstName;
            this.middleName = middleName;
            this.lastName = lastName;
        }

        public Student GetStudent()
        {
            return new Student(firstName, lastName, middleName, false, groupName);
        }
    }
}
```

Lecturer.cs

```
using System;


namespace DocsGenerator.Classes
{
    class Lecturer : Person
    {
        public Grade AcademicStatus;
        public string id;
        public ComType MemberType;
        public decimal MemberHours;


        public Lecturer(string Name, string Surname, string FatherName, Grade
AcademicStatus, string id = "") : base(Name, Surname, FatherName)
        {
            this.AcademicStatus = AcademicStatus;
            this.id = id;
        }


        public double GetMemberCost()
        {
            return GetAcademicCost() * GetMemberHours();
        }
        public double GetMemberHours()
        {
            return Decimal.ToDouble(MemberHours);
        }


        public double GetAcademicCost()
        {
            double academicCost = 0;


            switch (AcademicStatus)
            {
                case Grade.Assistant:
```

```
                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.Assistant);

                    break;

                case Grade.Docent:

                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.Docent);

                    break;

                case Grade.Prof:

                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.Prof);

                    break;

                case Grade.SeniorLecturer:

                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.SeniorLecturer);

                    break;

                case Grade.TechSciCand:

                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.TechSciCand);

                    break;

                case Grade.TechSciDoc:

                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.TechSciDoc);

                    break;

            }

            return academicCost;

        }

    }


    class LecturerJson

    {

        public string employeeId, divisionName, positionName, scientificRank,
asscientificDegree, firstName, middleName, lastName;

        public LecturerJson(string employeeId, string divisionName, string
positionName, string scientificRank, string asscientificDegree, string firstName,
string middleName, string lastName)

        {

            this.employeeId = employeeId;

            this.divisionName = divisionName;

            this.positionName = positionName;
```

```
            this.scientificRank = scientificRank;
            this.asscientificDegree = asscientificDegree;
            this.firstName = firstName;
            this.middleName = middleName;
            this.lastName = lastName;
        }
        public Lecturer GetLecturer()
        {
            return new Lecturer(firstName, lastName, middleName, Grade.Assistant,
employeeId);
        }
    }

    enum Grade
    {
        Assistant, SeniorLecturer, TechSciCand, Docent, TechSciDoc, Prof
    }
    enum ComType
    {
        Kerivnik, Konsultant, Recensent, KomMember, KomDir
    }
}
```

Discipline.cs

```
using System;
using System.Collections.Generic;


namespace DocsGenerator.Classes
{
    class Discipline
    {
        public string name;
        public Lecturer lect;
        public decimal Lab, Practice;
        public bool KP, RGR, KR, Ref, isExam, isZal;


        public Discipline(string name, Lecturer lect, bool KP, bool RGR, bool KR,
bool Ref, bool isExam, bool isZal, decimal Lab, decimal Practice)
        {
            this.name = name;
            this.lect = lect;
            this.KP = KP;
            this.RGR = RGR;
            this.KR = KR;
            this.Ref = Ref;
            this.isExam = isExam;
            this.isZal = isZal;
            this.Lab = Lab;
            this.Practice = Practice;


        }
        public double GetCost()
        {
            return GetCostKP() + GetCostKR() + GetCostExam() + GetCostLab() +
GetCostPractice() + GetCostRef() + GetCostRGR() + GetCostZal();
        }
        public double GetAcademicCost()
        {
            double academicCost = 0;
```

```
            switch (lect.AcademicStatus)
            {
                case Grade.Assistant:
                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.Assistant);
                    break;
                case Grade.Docent:
                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.Docent);
                    break;
                case Grade.Prof:
                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.Prof);
                    break;
                case Grade.SeniorLecturer:
                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.SeniorLecturer);
                    break;
                case Grade.TechSciCand:
                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.TechSciCand);
                    break;
                case Grade.TechSciDoc:
                    academicCost =
Decimal.ToDouble(Properties.Settings.Default.TechSciDoc);
                    break;
            }
            return academicCost;
        }
        public double GetCostKP()
        {
            return GetAcademicCost() * GetHoursKP();
        }
        public double GetCostKR()
        {
            return GetAcademicCost() * GetHoursKR();
        }
```

```
public double GetCostRGR()
{
    return GetAcademicCost() * GetHoursRGR();
}
public double GetCostRef()
{
    return GetAcademicCost() * GetHoursRef();
}
public double GetCostExam()
{
    return GetAcademicCost() * GetHoursExam();
}
public double GetCostZal()
{
    return GetAcademicCost() * GetHoursZal();
}
public double GetCostLab()
{
    return GetAcademicCost() * GetHoursLab();
}
public double GetCostPractice()
{
    return GetAcademicCost() * GetHoursPractice();
}
public double GetHoursKP()
{
    if (KP)
    {
        if (Process.GetInstance().GetStudentIsElderCourse())
        {
            return Decimal.ToDouble(Properties.Settings.Default.KPElder);
        }
        else
        {
            return Decimal.ToDouble(Properties.Settings.Default.KP);
```

```
        }
    }
    return 0;
}
public double GetHoursKR()
{
    if (KR)
    {
        if (Process.GetInstance().GetStudentIsElderCourse())
        {
            return Decimal.ToDouble(Properties.Settings.Default.KRElder);
        }
        else
        {
            return Decimal.ToDouble(Properties.Settings.Default.KR);
        }
    }
    return 0;
}
public double GetHoursRGR()
{
    if (RGR)
    {
        if (Process.GetInstance().GetStudentIsElderCourse())
        {
            return Decimal.ToDouble(Properties.Settings.Default.RGRElder);
        }
        else
        {
            return Decimal.ToDouble(Properties.Settings.Default.RGR);
        }
    }
    return 0;
}
public double GetHoursRef()
```

```
{
    if (Ref)
    {
        if (Process.GetInstance().GetStudentIsElderCourse())
        {
            return Decimal.ToDouble(Properties.Settings.Default.RefElder);
        }
        else
        {
            return Decimal.ToDouble(Properties.Settings.Default.Ref);
        }
    }
    return 0;
}
public double GetHoursExam()
{
    if (isExam)
    {
        if (Process.GetInstance().GetStudentIsElderCourse())
        {
            return Decimal.ToDouble(Properties.Settings.Default.ExamElder);
        }
        else
        {
            return Decimal.ToDouble(Properties.Settings.Default.Exam);
        }
    }
    return 0;
}
public double GetHoursZal()
{
    if (isZal)
    {
        if (Process.GetInstance().GetStudentIsElderCourse())
        {
```

```
                return Decimal.ToDouble(Properties.Settings.Default.ZalElder);
            }
            else
            {
                return Decimal.ToDouble(Properties.Settings.Default.Zal);
            }
        }
        return 0;
    }
    public double GetHoursLab()
    {
        if (Process.GetInstance().GetStudentIsElderCourse())
        {
            return Decimal.ToDouble(Properties.Settings.Default.LabElder * Lab);
        }
        else
        {
            return Decimal.ToDouble(Properties.Settings.Default.Lab * Lab);
        }
    }
    public double GetHoursPractice()
    {
        if (Process.GetInstance().GetStudentIsElderCourse())
        {
            return Decimal.ToDouble(Properties.Settings.Default.PrElder *
Practice);
        }
        else
        {
            return Decimal.ToDouble(Properties.Settings.Default.Pr * Practice);
        }
    }
    public double GetHours()
    {
        return GetHoursKP() + GetHoursKR() + GetHoursRef() + GetHoursRGR() +
GetHoursExam() + GetHoursZal() + GetHoursLab() + GetHoursPractice();
```

```
}

public double GetBasicHoursPractice()
{
    if (Process.GetInstance().GetStudentIsElderCourse())
    {
        return Decimal.ToDouble(Properties.Settings.Default.PrElder);
    }
    else
    {
        return Decimal.ToDouble(Properties.Settings.Default.Pr);
    }
}
public double GetBasicHoursLab()
{
    if (Process.GetInstance().GetStudentIsElderCourse())
    {
        return Decimal.ToDouble(Properties.Settings.Default.LabElder);
    }
    else
    {
        return Decimal.ToDouble(Properties.Settings.Default.Lab);
    }
}
public double GetBasicHoursKP()
{
    if (Process.GetInstance().GetStudentIsElderCourse())
    {
        return Decimal.ToDouble(Properties.Settings.Default.KPElder);
    }
    else
    {
        return Decimal.ToDouble(Properties.Settings.Default.Pr * Practice);
    }
}
```

```
public double GetBasicHoursKR()
{
    if (Process.GetInstance().GetStudentIsElderCourse())
    {
        return Decimal.ToDouble(Properties.Settings.Default.KRElder);
    }
    else
    {
        return Decimal.ToDouble(Properties.Settings.Default.KR);
    }
}
public double GetBasicHoursRGR()
{
    if (Process.GetInstance().GetStudentIsElderCourse())
    {
        return Decimal.ToDouble(Properties.Settings.Default.RGRElder);
    }
    else
    {
        return Decimal.ToDouble(Properties.Settings.Default.RGR);
    }
}
public double GetBasicHoursRef()
{
    if (Process.GetInstance().GetStudentIsElderCourse())
    {
        return Decimal.ToDouble(Properties.Settings.Default.RefElder);
    }
    else
    {
        return Decimal.ToDouble(Properties.Settings.Default.Ref);
    }
}
public double GetBasicHoursZal()
{
```

```
            if (Process.GetInstance().GetStudentIsElderCourse())
            {
                return Decimal.ToDouble(Properties.Settings.Default.ZalElder);
            }
            else
            {
                return Decimal.ToDouble(Properties.Settings.Default.Zal);
            }
        }
        public double GetBasicHoursExam()
        {
            if (Process.GetInstance().GetStudentIsElderCourse())
            {
                return Decimal.ToDouble(Properties.Settings.Default.ExamElder);
            }
            else
            {
                return Decimal.ToDouble(Properties.Settings.Default.Exam);
            }
        }
    }
    class DisciplineJson
    {
        private string sessionRegisterId, disciplineName, studyTypeName, studyTypeId,
studyGroupName;


        public DisciplineJson(string sessionRegisterId, string disciplineName, string
studyTypeName, string studyTypeId, string studyGroupName)
        {
            this.sessionRegisterId = sessionRegisterId;
            this.disciplineName = disciplineName;
            this.studyTypeName = studyTypeName;
            this.studyTypeId = studyTypeId;
            this.studyGroupName = studyGroupName;
        }
        public string GetDisciplineName()
```

```
        {
            return disciplineName;
        }
    }
}
```

Document.cs

```
using System;
using System.IO;


namespace DocsGenerator.Classes
{
    class IDocument
    {
        protected readonly string templatePath =
$"{Directory.GetCurrentDirectory()}\\templates";
        public virtual void Generate(string path, string studentName)
        {
            throw new AccessViolationException("The virtual function is not
redefined");
        }
    }
}
```

Contract.cs

```
using System;
using System.Windows.Forms;
using Xceed.Words.NET;

namespace DocsGenerator.Classes
{
    class Contract : IDocument
    {
        private readonly string targetName = "\\contract.docx";

        public override void Generate(string path, string studentName)
        {
            try
            {
                using (DocX documentMain = DocX.Load(path + $"\\{studentName}.docx"))
                {
                    using (DocX document = DocX.Load(templatePath + targetName))
                    {
                        document.ReplaceText("{StudentName}",
Process.GetInstance().GetStudentName());
                        document.ReplaceText("{CostTotal}",
Process.GetInstance().GetTotalCost().ToString());
                        document.ReplaceText("{HoursTotal}",
Process.GetInstance().GetTotalHours().ToString());
                        documentMain.InsertDocument(document, true);
                        documentMain.Save();
                    }
                }
            }
            catch (Exception e)
            {
               MessageBox.Show(e.Message, "Не вдалося створити контракт",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
```

```
}
```

Act.cs

```
using System;
using System.IO;
using System.Windows.Forms;
using Xceed.Words.NET;

namespace DocsGenerator.Classes
{
    class Act : IDocument
    {
        private readonly string targetName = "\\act.docx";
        private readonly string templateBlock = "\\act_block.docx";
        private readonly string templateEnd = "\\act_end.docx";
        public override void Generate(string path, string studentName)
        {
            try
            {
                using (DocX document = DocX.Load(templatePath + targetName))
                {
                    document.ReplaceText("{StudentName}",
Process.GetInstance().GetStudentName());
                    foreach (Discipline discipline in
Process.GetInstance().GetDisciplines())
                    {
                        using( DocX block = DocX.Load(templatePath + templateBlock))
                        {
                            document.InsertDocument(block, true);
                        }
                        document.ReplaceText("{DisciplineName}", discipline.name);
                        document.ReplaceText("{DisciplineCost}",
discipline.GetCost().ToString());
                        document.ReplaceText("{DisciplineHours}",
discipline.GetHours().ToString());
                    }
                    using( DocX end = DocX.Load(templatePath + templateEnd))
                    {
                        document.InsertDocument(end, true);
```

```
                    }
                    document.ReplaceText("{CostTotal}",
Process.GetInstance().GetTotalCost().ToString());


                    document.InsertParagraph().InsertPageBreakAfterSelf();
                    document.SaveAs(path + $"\\{studentName}.docx");
                }
            }
            catch (Exception e)
            {
                MessageBox.Show(e.Message, "Не вдалося створити контракт",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```

Calculation.cs

```
using System;
using System.IO;
using System.Windows.Forms;
using Xceed.Words.NET;

namespace DocsGenerator.Classes
{
    class Calculation : IDocument
    {
        private readonly string targetName = "\\calculation.docx";
        private readonly string templateBlock = "\\calculation_block.docx";
        private readonly string templateEnd = "\\calculation_end.docx";
        public override void Generate(string path, string studentName)
        {
            try
            {
                using (DocX documentMain = DocX.Load(path + $"\\{studentName}.docx"
))
                {
                    using (DocX document = DocX.Load(templatePath + targetName))
                    {
                        document.ReplaceText("{StudentName}",
Process.GetInstance().GetStudentName());
                        foreach (Discipline discipline in
Process.GetInstance().GetDisciplines())
                        {
                            double temp;
                            using (DocX block = DocX.Load(templatePath +
templateBlock))
                            {
                                document.InsertDocument(block, true);
                            }
                            document.ReplaceText("{DisciplineName}",
discipline.name);
                            temp = discipline.GetHoursLab();
                            if (temp > 0)
```

```
                            {
                                document.ReplaceText("{Lab_H}", temp.ToString());
                                document.ReplaceText("{Lab_C}",
discipline.GetCostLab().ToString());
                                document.ReplaceText("{Lab_CH}",
discipline.GetAcademicCost().ToString());
                            }
                            else
                            {
                                document.ReplaceText("{Lab_H}", "-");
                                document.ReplaceText("{Lab_C}", "-");
                                document.ReplaceText("{Lab_CH}", "-");
                            }

                            temp = discipline.GetHoursPractice();
                            if (temp > 0)
                            {
                                document.ReplaceText("{Pr_H}", temp.ToString());
                                document.ReplaceText("{Pr_C}",
discipline.GetCostPractice().ToString());
                                document.ReplaceText("{Pr_CH}",
discipline.GetAcademicCost().ToString());
                            }
                            else
                            {
                                document.ReplaceText("{Pr_H}", "-");
                                document.ReplaceText("{Pr_C}", "-");
                                document.ReplaceText("{Pr_CH}", "-");
                            }

                            temp = discipline.GetHoursKP();
                            if (temp > 0)
                            {
                                document.ReplaceText("{Kp_E}", "Є");
                                document.ReplaceText("{Kp_NE}", " Н");
                                document.ReplaceText("{Kp_C}",
discipline.GetCostKP().ToString());
```

```
                                document.ReplaceText("{Kp_CH}",
discipline.GetAcademicCost().ToString());
                            }
                            else
                            {
                                document.ReplaceText("{Kp_E}", " Є");
                                document.ReplaceText("{Kp_NE}", "Н");
                                document.ReplaceText("{Kp_C}", "-");
                                document.ReplaceText("{Kp_CH}", "-");
                            }

                            temp = discipline.GetHoursKR();
                            if (temp > 0)
                            {
                                document.ReplaceText("{Kr_E}", "Є");
                                document.ReplaceText("{Kr_NE}", " Н");
                                document.ReplaceText("{Kr_C}",
discipline.GetCostKR().ToString());
                                document.ReplaceText("{Kr_CH}",
discipline.GetAcademicCost().ToString());
                            }
                            else
                            {
                                document.ReplaceText("{Kr_E}", " Є");
                                document.ReplaceText("{Kr_NE}", "Н");
                                document.ReplaceText("{Kr_C}", "-");
                                document.ReplaceText("{Kr_CH}", "-");
                            }

                            temp = discipline.GetHoursRGR();
                            if (temp > 0)
                            {
                                document.ReplaceText("{Rgr_E}", "Є");
                                document.ReplaceText("{Rgr_NE}", " Н");
                                document.ReplaceText("{Rgr_C}",
discipline.GetCostRGR().ToString());
```

```
                                document.ReplaceText("{Rgr_CH}",
discipline.GetAcademicCost().ToString());
                            }
                            else
                            {
                                document.ReplaceText("{Rgr_E}", " Є");
                                document.ReplaceText("{Rgr_NE}", "Н");
                                document.ReplaceText("{Rgr_C}", "-");
                                document.ReplaceText("{Rgr_CH}", "-");
                            }

                            temp = discipline.GetHoursRef();
                            if (temp > 0)
                            {
                                document.ReplaceText("{Ref_E}", "Є");
                                document.ReplaceText("{Ref_NE}", " Н");
                                document.ReplaceText("{Ref_C}",
discipline.GetCostRef().ToString());
                                document.ReplaceText("{Ref_CH}",
discipline.GetAcademicCost().ToString());
                            }
                            else
                            {
                                document.ReplaceText("{Ref_E}", " Є");
                                document.ReplaceText("{Ref_NE}", "Н");
                                document.ReplaceText("{Ref_C}", "-");
                                document.ReplaceText("{Ref_CH}", "-");
                            }

                            temp = discipline.GetHoursZal();
                            if (temp > 0)
                            {
                                document.ReplaceText("{Zal_E}", "Є");
                                document.ReplaceText("{Zal_NE}", " Н");
                                document.ReplaceText("{Zal_C}",
discipline.GetCostZal().ToString());
```

```
                                document.ReplaceText("{Zal_CH}",
discipline.GetAcademicCost().ToString());
                            }
                            else
                            {
                                document.ReplaceText("{Zal_E}", " Є");
                                document.ReplaceText("{Zal_NE}", "Н");
                                document.ReplaceText("{Zal_C}", "-");
                                document.ReplaceText("{Zal_CH}", "-");
                            }

                            temp = discipline.GetHoursExam();
                            if (temp > 0)
                            {
                                document.ReplaceText("{Ex_E}", "Є");
                                document.ReplaceText("{Ex_NE}", " Н");
                                document.ReplaceText("{Ex_C}",
discipline.GetCostExam().ToString());
                                document.ReplaceText("{Ex_CH}",
discipline.GetAcademicCost().ToString());
                            }
                            else
                            {
                                document.ReplaceText("{Ex_E}", " Є");
                                document.ReplaceText("{Ex_NE}", "Н");
                                document.ReplaceText("{Ex_C}", "-");
                                document.ReplaceText("{Ex_CH}", "-");
                            }
                            document.ReplaceText("{Lab_N}",
discipline.GetBasicHoursLab().ToString());
                            document.ReplaceText("{Pr_N}",
discipline.GetBasicHoursPractice().ToString());
                            document.ReplaceText("{Kp_N}",
discipline.GetBasicHoursKP().ToString());
                            document.ReplaceText("{Kr_N}",
discipline.GetBasicHoursKR().ToString());
                            document.ReplaceText("{Rgr_N}",
discipline.GetBasicHoursRGR().ToString());
```

```
                            document.ReplaceText("{Ref_N}",
discipline.GetBasicHoursRef().ToString());

                            document.ReplaceText("{Zal_N}",
discipline.GetBasicHoursZal().ToString());

                            document.ReplaceText("{Ex_N}",
discipline.GetBasicHoursExam().ToString());



                            document.ReplaceText("{DisciplineCost}",
discipline.GetCost().ToString());

                            document.ReplaceText("{DisciplineHours}",
discipline.GetHours().ToString());

                        }

                        using (DocX end = DocX.Load(templatePath + templateEnd))

                        {

                            document.InsertDocument(end, true);

                        }

                        document.ReplaceText("{CostTotal}",
Process.GetInstance().GetTotalCost().ToString());

                        document.ReplaceText("{HoursTotal}",
Process.GetInstance().GetTotalHours().ToString());



                        documentMain.InsertDocument(document, true);

                        documentMain.InsertParagraph().InsertPageBreakAfterSelf();

                        documentMain.Save();

                    }


                }

            }

            catch (Exception e)

            {

                MessageBox.Show(e.Message, "Не вдалося створити контракт",
MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

    }

}
```

EK.cs

```
using System;
using System.Windows.Forms;
using Xceed.Words.NET;

namespace DocsGenerator.Classes
{
    class EK:IDocument
    {
        private readonly string targetName = "\\ek.docx";
        public override void Generate(string path, string studentName)
        {
            try
            {
                using (DocX document = DocX.Load(templatePath + targetName)) //С
каким названием должен генерироваться файл?
                {

                    document.ReplaceText("{StudentName}",
Process.GetInstance().GetStudentName());
                    document.ReplaceText("{CostTotal}",
Process.GetInstance().GetComissionTotalCost().ToString());
                    document.ReplaceText("{HoursTotal}",
Process.GetInstance().GetComissionTotalHours().ToString());

                    document.ReplaceText("{KerivnikCost}",
Process.GetInstance().GetMemberTypeCost(ComType.Kerivnik).ToString());
                    document.ReplaceText("{ConsultantCost}",
Process.GetInstance().GetMemberTypeCost(ComType.Konsultant).ToString());
                    document.ReplaceText("{ComDirCost}",
Process.GetInstance().GetMemberTypeCost(ComType.KomDir).ToString());
                    document.ReplaceText("{ComMemCost}",
Process.GetInstance().GetMemberTypeCost(ComType.KomMember).ToString());
                    document.ReplaceText("{ComRecenCost}",
Process.GetInstance().GetMemberTypeCost(ComType.Recensent).ToString());

                    document.ReplaceText("{Ker_H}",
Process.GetInstance().GetMemberTypeHours(ComType.Kerivnik).ToString());
```

```
                    document.ReplaceText("{Kons_H}",
Process.GetInstance().GetMemberTypeHours(ComType.Konsultant).ToString());

                    document.ReplaceText("{CDir_H}",
Process.GetInstance().GetMemberTypeHours(ComType.KomDir).ToString());

                    document.ReplaceText("{CMem_H}",
Process.GetInstance().GetMemberTypeHours(ComType.KomMember).ToString());

                    document.ReplaceText("{Rec_H}",
Process.GetInstance().GetMemberTypeHours(ComType.Recensent).ToString());



                    document.ReplaceText("{Ker_CH}",
(Process.GetInstance().GetMemberTypeCost(ComType.Kerivnik) /
Process.GetInstance().GetMemberTypeHours(ComType.Kerivnik)).ToString());

                    document.ReplaceText("{Kons_CH}",
(Process.GetInstance().GetMemberTypeCost(ComType.Konsultant) /
Process.GetInstance().GetMemberTypeHours(ComType.Konsultant)).ToString());

                    document.ReplaceText("{CDir_CH}",
(Process.GetInstance().GetMemberTypeCost(ComType.KomDir) /
Process.GetInstance().GetMemberTypeHours(ComType.KomDir)).ToString());

                    document.ReplaceText("{CMem_CH}",
(Process.GetInstance().GetMemberTypeCost(ComType.KomMember) /
Process.GetInstance().GetMemberTypeHours(ComType.KomMember)).ToString());

                    document.ReplaceText("{Rec_CH}",
(Process.GetInstance().GetMemberTypeCost(ComType.Recensent) /
Process.GetInstance().GetMemberTypeHours(ComType.Recensent)).ToString());



                    document.ReplaceText("{Ker_C}",
Process.GetInstance().GetMemberTypeCost(ComType.Kerivnik).ToString());

                    document.ReplaceText("{Kons_C}",
Process.GetInstance().GetMemberTypeCost(ComType.Konsultant).ToString());

                    document.ReplaceText("{CDir_C}",
Process.GetInstance().GetMemberTypeCost(ComType.KomDir).ToString());

                    document.ReplaceText("{CMem_C}",
Process.GetInstance().GetMemberTypeCost(ComType.KomMember).ToString());

                    document.ReplaceText("{Rec_C}",
Process.GetInstance().GetMemberTypeCost(ComType.Recensent).ToString());



                    document.ReplaceText("{ComissionCost}",
Process.GetInstance().GetComissionTotalCost().ToString());



                    document.InsertParagraph();

                    document.SaveAs(path + $"\\{studentName}.docx");

                }

            }
```

```
            catch (Exception e)
            {
                MessageBox.Show(e.Message, "Не вдалося створити контракт",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```